Modeling the Heat of Formation of Organic Compounds Using SPARC

by

Tad S. Whiteside

(Under the direction of Lionel Carreira)

Abstract

Typically, the interaction of chemicals with the environment is governed through physicochemical properties. The Environmental Protection Agency has developed several models to predict the fate of chemicals in the environment. SPARC (SPARC Performs Automated Reasoning in Chemistry) has been developed as a method to predict the properties of environmentally sensitive compounds.

SPARC uses computational algorithms based on chemical structure theory to calculate chemical properties, including the heat of formation. Molecular structures are broken into simple functional units (reactophores) with intrinsic properties. Each reactophore is analyzed and the effects of appended molecular structures are quantified through perturbation theory.

Standard enthalpies of formation (ΔH_f) were calculated with models developed using the computer program SPARC. The ΔH_f models have been completely developed using all known data for saturated and unsaturated hydrocarbons and halogenated hydrocarbons. Basic models have also been developed for alcohols, aldehydes, and ketones. The structures of these compounds vary from chains and conjugated rings to poly-benzoic aromatic hydrocarbons. The 587 hydrocarbons have a SPARC calculated RMS of 4.50 kJ mol^{-1}. Halogenated hydrocarbons have a calculated RMS deviation of 5.18 kJ mol^{-1} for 202 compounds.

The effect of stereochemistry on the standard enthalpy of formation was also modeled. Chiral centers are found in a variety of molecules and help define the overall structure of a compound. The local atomic environment determines the strain energy in each chiral center. There are four local environments in which chiral centers are found: Linear, Single, Bridge, and SideShare. These are modeled independently and the total contribution of stereochemistry to the heat of formation is determined by summing the energy found in these environments. The 169 experimentally determined compounds with chiral centers were used to develop this model.

To provide a benchmark of SPARC's capabilities, the heat of formation of the compounds used to develop the models was also calculated using the semi-empirical PM3 method and the group additivity method developed by Benson, as implemented by NIST in their chemical webbook. SPARC outperforms both of these methods in terms of speed and accuracy.

INDEX WORDS: enthalpy of formation, heat of formation, hydrocarbons, halohydrocarbons, SPARC, stereochemistry

Modeling the Heat of Formation of Organic Compounds Using SPARC

by

Tad S. Whiteside

B.S., Erskine College, 2000

B.A., Erskine College, 2000

A Dissertation Submitted to the Graduate Faculty

of The University of Georgia in Partial Fulfillment

of the

Requirements for the Degree

Doctor of Philosophy

Athens, Georgia

2004

Modeling the Heat of Formation of Organic Compounds Using SPARC

by

Tad S. Whiteside

Dedication

To my family

Acknowledgments

I would like to thank Dr. L. Carreira for his support and encouragement over the past four years. I appreciate the time and effort he has expended on my behalf. I would also like to thank the members of my committee, Dr. J. Amster and Dr. M. Duncan, for providing sound advice.

I appreciate Dr. S. Hilal's insight into the heat of formation models and SPARC in general. I also give special thanks to Raj Ayyampalayam for allowing me to be impatient and short-tempered.

This research was sponsored in part by the U.S. Environmental Protection Agency under the cooperative agreement No. CR83138401 with the University of Georgia.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Introduction

As more countries enter the global marketplace, the potential for environmental damage from unregulated dumping and industrial waste increases. While most countries have environmental monitoring and regulating agencies, these are typically under-funded and under-staffed. With minimal support, these agencies attempt to monitor the effects thousands of various chemicals have on the environment. Typically, the interaction of chemicals with the environment is governed through physicochemical properties. It has been common practice for these agencies to determine these properties by analyzing waste and compounds using traditional methods such as field monitoring, toxicological testing, and laboratory assessments. A separate procedure is usually necessary to determine the value of each property for a given chemical. In order to fully determine these values and because the chemicals of interest are generally toxic, a large amount of time must be invested in analysis, disposal, and safety. With the explosive growth of chemical usage in all industries and with over 70,000 chemicals listed in the U.S. EPA's Office of Toxic Substances database[1] alone, for these agencies to attempt to experimentally determine each property is an impractical, if not impossible, undertaking.

In order to provide regulatory guidelines, several engineering models have been developed which predict the fate of compounds in the environment.[1] While these models are able to predict the fate of compounds in the environment, they depend on reactivity parameters: values only obtained from the physical and chemical constants associated with kinetic and thermodynamic descriptions of reactivity. Unfortunately, the properties of many environmentally important chemicals are either unknown or incomplete. However, many of these properties have been successfully modeled using computer programs. These property calculators have been developed to the extent that accurate and rapid estimation for certain classes of compounds is easily accomplished.[2, 3] The advantage of these calculators is that the time spent on property determination is reduced to minutes instead of months and the disposal and safety issues become moot.[1] SPARC is one such calculator, but differs from the others because instead of modeling a class of compounds, SPARC attempts to model the chemical effects which contribute to each property.

1.2 SPARC

SPARC (SPARC Performs Automated Reasoning in Chemistry) is a computer program developed to analyze molecular properties and reactions by applying the reasoning process of an expert organic chemist.[1] SPARC does not calculate chemical properties from "first principles". Instead, it acts as an intelligent agent that analyzes the chemical structure of a compound and based on the structure and property being calculated, applies interaction models developed for that property. The properties SPARC calculates are based on "real" molecules found in "real environments". The same basic method and general equations are used for each property SPARC models. This includes using SPARC's structural analysis engine to factor molecules into individual functional groups (reaction centers and perturbers) and then determining, for the property of interest, how each factored group affects the others in contributing to the overall property. The reaction centers, or reactophores, are the smallest subunits of the larger molecule with known chemical properties.[1] The perturbers are molecular structures attached to the reactophore which affect its energy.

The basic quantity of each reaction center is a measured, unperturbed, value. If the reaction center is the only group in the molecule, there are no perturbing structures; hence, the value for the property is the value in SPARC's database of measured values. Otherwise, the reactophore's value will differ from the measured value by the sum of the perturbations caused by the appended structures. These perturbations are quantified into the various interactions commonly occurring in physical organic chemistry such as steric, resonance, connectivity, etc. While each reaction center is analyzed by a common method, the results will differ depending on the reaction center, groups connected to the reaction center, and the property being determined. After the value of each reactophore is determined, the entire molecule is analyzed to account for any additional structural or resonance effects impacting the property of interest. These values are summed together to produce the SPARC calculated value for the property of interest. The SPARC method of calculating physical and chemical properties is fast and accurate. For many of these properties, the calculated value deviates from the experimentally accepted value by an amount on the order of the error range reported for the experiment.[1]

1.3 The Heat of Formation

The heat of formation or standard enthalpy of formation (ΔH_f), is the standard reaction enthalpy for the formation of a compound from its component elements in their reference states.[4] The reference state is the most stable state at the specified temperature and 1 bar.[4] The enthalpy of formation is one of the basic thermodynamic properties of molecules. One application of this property is in examining the reactivities of compounds to aid in the prediction of equilibrium constants.[5] We are interested in using the heat of formation models to aid in the development of reaction kinetics and reduction-oxidation models. These models will improve the prediction of transition state energies and provide a better understanding of reaction pathways.

1.4 Calculating the Heat of Formation

Currently, heats of formation are calculated primarily through two methods, *ab initio* calculations and the group additivity method developed by Benson. While other methods, including molecular mechanics and semi-empirical PM3 calculations, are also used, these are not as prevalent as the *ab initio* or group additivity methods. The SPARC method for determining the heat of formation is not an attempt to take away from these other methods but is a stepping stone to the development of an improved kinetics model.

Ab initio methods calculate the heat of formation for a molecule from its component atoms. One method first calculates the ΔE at 0K using the optimized geometries of the compound formed from its gas phase elements.[6] This energy change is then converted to ΔH_f at 298K by adding the translational, rotational, and vibrational contributions to the energy through the ideal gas model, and finally applying the definition $H = E + PV = E + RT$.[6] The calculation is refined by adding correction terms based on the known heats of formation for a set of compounds varying in composition and structure.[6] This method works well for many molecules; however, it is computationally expensive and the accuracy rapidly deteriorates after a molecular size of approximately 10 atoms is exceeded. The error reported for a typical calculation is 16 kJ mol^{-1} and the time to complete a calculation is measured in tens of hours.[6]

The estimation method developed by Benson is both simple and accurate. Benson's approach is based on the premise that most properties of larger molecules can be considered as the sum of the contributions from the individual atoms or bonds in the molecule.[5, 7] Such a proposal is reasonable because it has been shown that forces between atoms are highly dependant upon distance.[1, 5, 7] Benson extends this proposal from individual atoms to groups of atoms. Using the component groups of a molecule, the property in question can be calculated by summing the contributions from each group.[5, 7] Benson derives the group data from the observed heats of formation of known compounds.[5, 7] This method does a much better job of calculating the heats of formation in terms of both speed and accuracy when compared with *ab initio* calculations. The usual estimation error is 8 kJ mol^{-1} and the time to calculate is measured in minutes.[5, 7]

The SPARC method for calculating the heat of formation incorporates strategies from both of these methods. The model also uses procedures developed for other SPARC models. The heat of formation can be expressed as a function of the energy required to add an atom to the free element state. It represents the energy difference between the free element state and the whole molecule state. To model this energy difference, ΔH_f is expressed in terms of the summation of the contributions of all the components, perturber(s), and reaction centers(s), in the molecule (Equation 1.1).

$$\Delta H_f = \sum_{C=1}^{n} [(\Delta H_f)_C + A_C] + (\Delta H_f)_{Whole\_molecule} \tag{1.1}$$

Using the standard SPARC procedure, molecules are first broken into C reaction centers (essential atoms/groups) and their perturber structures. In this equation, $(\Delta H_f)_C$ is the intrinsic heat of formation of the reaction center, C, and is assumed to be unperturbed and independent of any perturbing structures. A_C describes adjustments made to the $(\Delta H_f)_C$ by the perturber structure(s); the $(\Delta H_f)_{Whole\_molecule}$ is any additional energy added (or subtracted) due to the overall molecular structure, e.g. distribution of NBMO charge in conjugated systems.

The reaction center perturbation, A_C, is further factored into mechanistic components SPARC calculates for perturbations to the heat of formation of each reaction center, as described by Equation 1.2.

$$A_C = A_{steric}(\Delta H_f)_C + A_{resonance}(\Delta H_f)_C + A_{connectivity}(\Delta H_f)_C \tag{1.2}$$

$A_{steric}(\Delta H_f)_C$, $A_{resonance}(\Delta H_f)_C$, and $A_{connectivity}(\Delta H_f)_C$ describe how the size, resonance, and connectivity of the perturber(s) affect the reaction center's value. The steric interaction is the effect of a perturber's size on the reaction center. The resonance interaction is a description of the amount of stabilization a molecule undergoes due to charge being distributed out of the reaction center and into the perturbing structures. The connectivity adjustment describes the change in a reactophore's energy when a hydrogen atom of a base reactophore is replaced with another group.

The "whole molecule" correction, $(\Delta H_f)_{Whole\_molecule}$, is the sum of corrections applied to the entire molecule; this includes corrections for rings, additional resonance, and additional steric energy (Equation 1.3).

$$(\Delta H_f)_{Whole\_molecule} = (\Delta H_f)_{rings} + (\Delta H_f)_{xres} + (\Delta H_f)_{xsteric} \tag{1.3}$$

The number, size, and connectivity of the rings in a molecule determine the amount of correction for rings, $(\Delta H_f)_{rings}$. There is an additional resonance correction, $(\Delta H_f)_{xres}$, for homoaromaticity. $(\Delta H_f)_{xsteric}$ models the strain induced by various spatial environments. All of the perturbations and adjustments are empirically trained against experimentally determined values of the heat of formation.

SPARC's method to calculate the ΔH_f has fewer parameters when compared to Benson's method. The Benson method models compounds by assigning every possible group a value; e.g. both C-Cl and C(Cl)(Br) have separate values. This leads to large numbers of parameters that must be determined before a calculation can be completed. With SPARC, instead of modeling a complete group, such as a C-Cl group, just the methyl group is modeled and the chlorine group is modeled. The steric, electrostatic, and other effects of that group are modeled so when the two groups are combined, the correct contribution from each model is added to the total and the answer is produced.

1.5 Structure of Dissertation

The second, third, and fourth chapters have been published or submitted for publication in the Journal of Theoretical and Computational Chemistry and the Journal of Chemical Theory and Computation. The second and third chapters explain the heat of formation model for hydrocarbons and halohydrocarbons, respectively. Chapter four describes how stereochemistry affects the heat of formation. The fifth chapter includes the work done on ketones, aldehydes, and alcohols and the sixth chapter explains the tools developed to expedite model development.

1.6 Data

The experimental values used in constructing the model come from References [8]-[10]. The types of experiments used to produce the data are: static bomb calorimetry, rotating bomb calorimetry, calorimetry of hydrogenation, heat of isomerization, heat of equilibrium, kinetic measurements, and ion cycles.[8] When more than one method of obtaining the heat of formation for the molecule of interest was reported, either the data reported from bomb calorimetry or, lacking that, the best reported value was used as the basis for building the model. Unfortunately, the NIST website[8] has inconsistencies and large errors with some of the reported experimental values. Other problems in building the model occurred when more than one experimental value was reported for a compound and these differed by a significant amount. When discrepancies were discovered, the original publications as well as values reported by other groups were examined to determine the validity of the data. The value selected was the one reported by the researcher whose publication was most clearly documented.

Chapter 2

Prediction of the Enthalpy of Formation of Hydrocarbons Using SPARC[1]

[1]Whiteside, T.S. and L.A. Carreira. *J. Theoretical and Computational Chemistry*. World Scientific: Singapore, 2004; 3:451–469. Reprinted here with permission of publisher.

2.1 Introduction

The enthalpy of formation (heat of formation or ΔH_f) is one of the basic thermodynamic properties of molecules. Scientists use this property in many diverse applications; however, the most practical applications are in examining the reactivities of compounds and aiding in the prediction of equilibrium constants.[1] We are interested in using the heat of formation models to aid in the development of reaction kinetics and reduction-oxidation models to improve the prediction of transition state energies. For a large number of compounds, the heat of formation and other physicochemical properties are unknown. To experimentally determine these properties for every substance would be prohibitively expensive in terms of both time and effort. Fortunately, several methods have been developed to calculate the enthalpy of formation and related physical properties. Previously this has been accomplished through *ab initio*, semi-empirical, and group additivity methods. Recently, SPARC models have been developed and used to calculate many physical properties of organic compounds including the Henry's constant, vapor pressure, boiling point, solubility, hydrolysis rates, aqueous and nonaqueous pKa's, reduction potential, index of refraction, and now, heats of formation.[2, 3, 4, 5, 6]

SPARC (SPARC Performs Automated Reasoning in Chemistry) is a computer program developed to analyze molecular properties and reactions in the same manner an expert chemist would. For each chemical property SPARC uses the same basic method and general equations. This includes using SPARC's structural analysis engine to factor molecules into individual functional groups (reaction centers and perturbers) and then determining, for the property of interest, how each factored group affects the others in contributing to the overall property. The reaction centers, or reactophores, are the smallest subunits of the larger molecule with known chemical properties.[5] The perturbers are molecular structures attached to the reactophore that affect the value of the reactophore. The basic quantity for each reaction center is a measured, unperturbed, value. If the reaction center is the only group in the molecule, there are no perturbing structures; hence, the value for the property is the value in SPARC's database of measured values. Otherwise, the reactophore's value will differ from the measured value by the sum of the perturbations caused by the appended structures. These perturbations are quantified into the various interactions commonly occurring in physical organic chemistry such as steric, resonance, connectivity, etc. Each reaction center is analyzed by a common method; however, the results of the analysis will differ depending on the reaction center, groups connected to the reaction center, and the property being determined. After each reactophore's value has been determined, the entire molecule is analyzed to account for any additional structural or resonance effects that may impact the property of interest. All of these values are summed together to produce the SPARC calculated value for the property of interest. For a complete description of the SPARC method please see reference [5].

As stated in reference [1], at room temperature, a shift in the enthalpy of reaction by 1 kcal mol^{-1} (4.184 kJ mol^{-1}) will generally change the equilibrium constant by a factor of five. This is the difference in a 90% and a 64% reaction yield; or, in terms of time, has the effect of multiplying the reaction time by five.[1] This change applies equally to reactions whether the total enthalpy of reaction is 20 kJ mol^{-1} or 2000 kJ mol^{-1}.[1] Therefore, for any calculation scheme to be accurate within a factor of ten, its uncertainty must be less than 9 kJ mol^{-1}. It was our goal to develop a heat of formation calculator that would meet or exceed these requirements.

2.2 Theoretical Methodology

2.2.1 Calculation of the ΔH_f Using SPARC

The calculation of the heat of formation uses procedures developed for other SPARC models. As these have already been presented in previous publications[2, 3, 4, 5], only the methods and models specifically developed for determining the heat of formation will be presented here. Using standard SPARC procedure, molecules are first broken into C reaction centers (essential atoms/groups) and their perturber structures; the heat of formation is then calculated using Equation 2.1. In this equation, $(\Delta H_f)_C$ is the intrinsic heat of formation of the reaction center, C, and is assumed to be unperturbed and independent of any perturbing structures. A_C describes adjustments made to the $(\Delta H_f)_C$ by the perturber structure(s); the $(\Delta H_f)_{Whole\_molecule}$ is any additional energy added (or subtracted) due to the overall molecular structure, e.g. distribution of NBMO charge in conjugated systems.

$$\Delta H_f = \sum_{C=1}^{n} [(\Delta H_f)_C + A_C] + (\Delta H_f)_{Whole\_molecule} \tag{2.1}$$

The reaction center perturbation, A_C, is further factored into the mechanistic components SPARC calculates for perturbations to the heat of formation of each reaction center, as described by Equation 2.2. Here $A_{steric}(\Delta H_f)_C$,$A_{resonance}(\Delta H_f)_C$, and $A_{connectivity}(\Delta H_f)_C$ describe how the size, resonance, and connectivity of the perturber(s) affect the reaction center's value. The steric interactions are dependent on the size of the locally appended structure(s). The resonance interaction is a description of the amount of stabilization a molecule undergoes due to charge being distributed out of the reaction center and into the perturbing structures. The connectivity adjustment describes the change in a reactophore's energy when a hydrogen atom of a base reactophore is replaced with another group.

$$A_C = A_{steric}(\Delta H_f)_C + A_{resonance}(\Delta H_f)_C + A_{connectivity}(\Delta H_f)_C \tag{2.2}$$

The "whole molecule" correction, $(\Delta H_f)_{Whole\_molecule}$, is the sum of corrections applied to the entire molecule; this includes corrections for rings, additional resonance, and additional steric energy (Equation 2.3). The number, size, and connectivity of the rings in a molecule determines the amount of correction for rings. There is an additional resonance correction for homoaromaticity. The additional steric correction is for the gauche interaction between two reaction centers. All of the perturbations and adjustments are empirically trained against experimentally determined values of the heat of formation. Other descriptors, e.g. hydrogen bonding, sigma induction, etc. will become necessary when expanding these models beyond the first step of creating a hydrocarbon ΔH_f calculator.

$$(\Delta H_f)_{Whole\_molecule} = (\Delta H_f)_{rings} + (\Delta H_f)_{xres} + (\Delta H_f)_{xsteric} \tag{2.3}$$

2.2.2 Molecular Functional Groups (Reaction Centers)

When given a compound, SPARC first determines the constituent pieces of the molecule. For example, given the compound 3-amino-5-chloro-benzoic acid, SPARC would find four functional groups: a benzene group, a carboxylic group, an amine group, and a chlorine group. Depending on the property being modeled, the functional groups considered reaction centers and how they are used, will vary. For the heat of formation, all functional groups are considered reaction centers and each are analyzed for their contribution to the overall energy.

For the hydrocarbon model, SPARC recognizes four types of functional groups: methyl, ethylenic, acetylenic, and aromatic reaction centers. The $(\Delta H_f)_C$ values of the methyl, ethylenic, and acetylenic groups are the experimentally determined values of the heat of formation for methane, ethylene, and acetylene respectively; these values are stored in the SPARC database. The fourth type of reaction center, aromatic, is not a simple tabulated value, but is calculated and depends on the exact type of reaction center. Each reaction center is possibly perturbed through steric, connectivity, or resonance interactions.

Methyl Reaction Centers

Methyl reaction centers are carbon atoms connected to other atoms only through sp3 bonds. Their $(\Delta H_f)_C$ value is simply the literature value of methane, -74.6 kJ mol^{-1}. The methyl reaction center's value can be perturbed through both steric and connectivity effects.

Ethylenic Reaction Centers

Ethylenic reaction centers are two carbon atoms connected to each other by a double bond and to other atoms by single bonds. Their $(\Delta H_f)_C$ value is the literature value of ethylene,

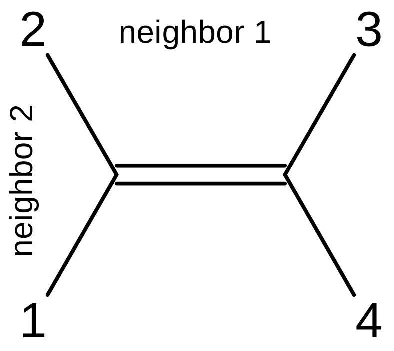

Figure 2.1: The ethylenic reaction center can experience interactions at four locations

52.47 kJ mol^{-1}. Ethylenic reaction centers are difficult to model due to the possibility of four steric interactions on each reaction center (Figure 2.1). Other perturbations include connectivity, resonance, and overall molecular structure (e.g. if the reaction center is endo or exo in a ring).

For ethylenic reaction centers inside a ring, it becomes necessary to analyze the reaction center perturbations differently. The first perturbation to determine is the one describing how the reaction center is affected by being inside a ring. This perturbation takes into account the steric and connectivity perturbations the ring has on the reaction center. It includes variables such as the size of the ring, the location of the reaction center inside the ring, and in what other rings the reaction center is included. The location inside the ring is important because this determines if the ring-atom substituents are *cis* or *trans* to each other (Figure 2.2). The other perturbations to account for are those between substituents not members of the ring and the interaction of these with the ring and with each other (Figure 2.3).

Exo ring ethylenic reaction centers have a perturbation based on the size of the ring. Also, substituents off of the sp2 carbon, which are not part of the ring, cause additional perturbations to the reaction center due to steric and connectivity effects. These perturbations are dependant on the number and size of substituents (Figure 2.4). For two rings joined by

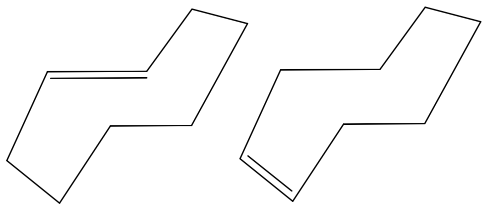

Figure 2.2: trans- and cis- cyclooctene

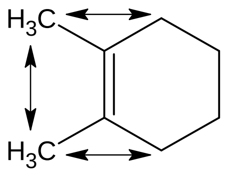

Figure 2.3: Interactions in 1,2-dimethyl cyclohexane

an exo double bond (Figure 2.5) the only perturbation to the ethylenic reaction center is due to the size of each ring.

Acetylenic Reaction Centers

Acetylenic reaction centers are two carbon atoms connected to each other with a triple bond, and to other atoms through a single bond. Their $(\Delta H_f)_C$ value is the literature value of acetylene, 227.40 kJ mol^{-1}. Steric and connectivity perturbations also affect acetylenic

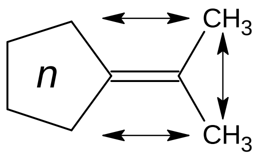

Figure 2.4: Possible interactions between substituents and an n-membered ring in an exo-ethylenic reaction center.

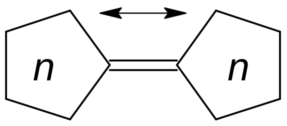

Figure 2.5: Interaction between two n-membered rings joined by an exo double bond.

reaction centers. For acetylenic reaction centers in a ring, a further perturbation is necessary due to the additional strain of being bent from a linear geometry. Acetylenic reaction centers, like ethylenic and aromatic centers, can have resonance perturbations, although to a lesser extent.

Aromatic Reaction Centers

Aromatic reaction centers are those composed entirely of benzene subgroups and are more commonly known as poly-benzoic aromatic hydrocarbons (PBAHs). Initially, the values of all known PBAH's were tabulated and SPARC referenced them when necessary. However, only a few (twelve) have been accurately measured and a method to calculate the ΔH_f from structure, for any arbitrary PBAH, was needed. It was determined these values could be calculated to within experimental accuracy from just five parameters, so this calculational scheme replaced the tabulated values. The ΔH_f of the aromatic reaction center can be determined from these parameters and just three pieces of data: how the atoms in the molecule are connected to each other, the total number of phenyl rings in the molecule, and how these rings are connected to each other (Equation 2.4).

$$\begin{aligned}\Delta H_f = \\ &\sum_{C=1}^{n} [\delta_{sp2}(\Delta H_f)_C + \delta_{bridge}(\Delta H_f)_C + \delta_{buried}(\Delta H_f)_C] + \\ &(\Delta H_f)_{bent} + (\Delta H_f)_{resonance} + (\Delta H_f)_{helicity} \qquad (2.4)\end{aligned}$$

The ΔH_f of the reaction center is the sum of the contributions of each atom, C, and the overall correction based on the structure of the molecule. The $\delta_{sp2}(\Delta H_f)_C$ is the basic amount

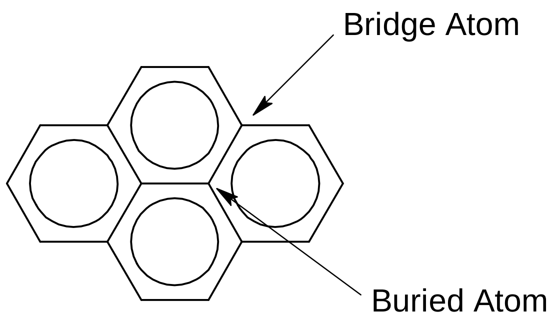

Figure 2.6: Pyrene, which has examples of bridge and buried atoms.

of energy contributed by an sp2 carbon atom. The $\delta_{bridge}(\Delta H_f)_C$ and $\delta_{buried}(\Delta H_f)_C$ are additional perturbations necessary if C is a 'bridge' or 'buried' carbon. A simple illustration of 'bridge' and 'buried' carbons is given by pyrene (Figure 2.6). The $(\Delta H_f)_{bent}$ correction is necessary when a molecule has a bent structure, like phenanthrene. A "bend" occurs when two benzenoids are "meta" to each other through a third benzenoid ring. On this third benzenoid, there are two hydrogen (or other non-sp2) atoms on the outside edge compared to none on the inside edge. This leads to a lowering of energy in the molecule. To find the total number of bends in a molecule, all such combinations are found. For example, in coronene, the system finds six bends; molecules like phenanthrene, where the outside edges are not connected to additional sp2 atoms, have one bend. In molecules where the outside edge of the bend is connected to additional sp2 atoms, one-half of a bend correction is contributed. Triphenylene would have 3×0.5 or 1.5 bend contributions. The total number of bends, calculated as described, are found and divided by the number of phenyl rings in the molecule to determine the entire bend contribution.

The $(\Delta H_f)_{resonance}$ adjustment scales as the number of phenyl rings in the PBAH. As the number of phenyl rings increases, the amount of resonance increases, lowering the overall energy.

Table 2.1: Calculating the heat of formation of pyrene using the SPARC parameters.

ΔH_f= AromaticAtoms * Val1	ΔH_f=(10+4+2) * 13.82
+ BridgeAtoms * Val2	+ 4 * 21.55
+ BuriedAtoms * Val3	+ 2 * 13.39
+ (NumBends/NumPhenylRings) * Val4	+ (2/4) * -89.73
+ NumPhenylRings * Val5	+ 4 * -15.89
+ NumHelical * Val6	+ 0 * 33
Observed= 225.7 kJ mol^{-1}	Calculated= 225.68 kJ mol^{-1}

The $(\Delta H_f)_{helicity}$ correction is required for molecules, such as benzo[c]phenanthrene, which are helical in overall structure. A molecule can be said to have helicity when there are at least four phenyl rings connected such that only one bond separates every other phenyl ring and these four phenyl rings do not share a common fifth phenyl ring as in benzo[ghi]perylene. When this occurs, the heat of formation is raised by approximately 33 kJ mol^{-1} for each set of four phenyl rings. This is most likely due to a reduction of the π-electron overlap in these rings because the overall molecule is no longer planar. This non-planarity may be enhanced by the repulsion of the atoms off of the 1 and 12 positions.

A sample calculation for determining the heat of formation of the reaction center pyrene (Figure 2.6) is shown in Table 2.1. This is an accurate and useful approach to predicting the heat of formation of aromatic groups. With this method, there are fewer parameters in the database and PBAHs without measured ΔH_f values can still be used to predict the heat of formation of the aromatic substructure. The RMS error using this scheme is within experimental accuracy, 2.35 kJ mol^{-1}. Table 2.2 has a complete listing of all the PBAHs for which the heat of formation was calculated. It should be noted that only after this model was developed it was discovered Benson had already described a similar mechanism for calculating the heat of formation of PBAHs.[1, 7]

Steric and connectivity perturbations also affect aromatic reaction centers. Determining the effect of each is somewhat more difficult than in the other reaction centers. This is

Table 2.2: Comparison of experimental[1] and calculated values of ΔH_f (kJ mol^{-1}) for PBAHs using SPARC, PM3[2] , and Group Additivity[3].

Name	**CAS Number**	**Observed Value**	**SPARC**	**PM3**	**Group Additivity**
Benzene	71-43-2	82.93	82.93	97.45	82.8
Naphthalene	91-20-3	150.6	150.00	169.13	151
Anthracene	120-12-7	230.8	228.49	256.58	218
Pyrene	129-00-0	225.7	226.53	266.81	231
Phenanthrene	85-01-8	201.2	203.88	228.81	209
Chrysene	218-01-9	269.8	270.07	294.67	268
Benz[a]anthracene	56-55-3	293	288.52	309.77	277
Tetracene	92-24-0	302.5[4]	306.98	351.08	286
Triphenylene	217-59-4	278.2[4]	279.29	283.88	273
Perylene	198-55-0	328[4]	327.17	341.24	280
Benzo[c]phenanthrene	195-19-7	291.2[4]	290.07	340.10	263
Coronene	191-07-1		348.56	363.83	324
Benzo[g,h,i]perylene	191-24-2		339.98	346.96	302
Benzo[a]pyrene	50-32-8		305.02	339.81	290
Benzo[e]pyrene	192-97-2		305.02	318.40	280
Dibenz[a,c]anthracene	215-58-7		363.32	361.48	326
Dibenz[a,h]anthracene	53-70-3		355.94	365.73	336
Ovalene	190-26-1		515.95	497.62	417
Anthanthrene	191-26-4		327.67	386.39	311
Pentacene	135-48-8		385.47	449.26	354
Dibenzo[a,,j]anthracene	224-41-9		355.94	365.79	336
dibenzo[a,l]pyrene	191-30-0		403.51	453.92	334
dibenzo[a,h]pyrene	189-64-0		383.51	418.54	348
dibenzo[a,i]pyrene	189-55-9		383.51	407.78	348
quaterrylene	188-73-8		678.82	683.06	539.7
dibenzo[a,e]pyrene	192-65-4		383.51	388.80	339
RMS		1.56	2.27	29.43	18.86

1 - all values from reference [19] unless otherwise noted

2 - PM3 values calculated using reference [15]

3 - Group Additivity method uses Benson's parameters [16]

4 - Welsh,Tong,et al. Thermochimica Acta, 290 (1996) 55-64

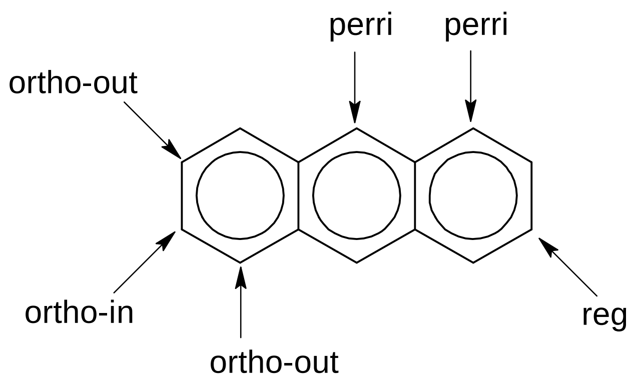

Figure 2.7: The steric interaction sites on an aromatic reaction center.

because there are four unique steric interaction sites on these hydrocarbons (Figure 2.7). The resonance perturbation between aromatic compounds is discussed in the resonance section.

2.3 ΔH_f Adjustment Models

The modeling of energy contributions can be broken into two sections: reaction center perturbations and whole molecule contributions. The reaction center perturbations are the sum of the effects the perturber structures have on one reaction center; the whole molecule effects are those that affect the energy of the entire molecule.

2.3.1 Reaction Center Models

Each reaction center has a basic value; this basic value can be perturbed by appended structures. These perturbations include steric, resonance, and connectivity.

Perturber Models: Steric

Steric perturbations can play a major role in affecting the energy of reaction centers. The function of this interaction is dependant on the reaction center being perturbed as well as the size of the appended structures. To determine the size of each structure, the conical volume is determined.[5] The volume of methane is then subtracted from this volume to describe the effective size of the structure. Structures with a volume smaller than methane will not have a steric contribution.

Methyl Steric The methyl reactophore has one unique substituent location and is tetrahedral in geometry. Since each substituent can affect the others, the number (n) and size (x_n) of the substituents found determines the total steric adjustment (Equation 2.5). ρ_{steric} quantifies the effect these substituents have on the reaction center and each other.

$$A_{steric}(\Delta H_f)_C = \rho_{steric}(n)x_n \tag{2.5}$$

Ethylenic Steric The ethylenic reaction center has one unique substituent location with four possible substituent interactions (Figure 2.1). Since each substituent can affect up to two others, the interactions must be broken into two parts (Equation 2.6). $\rho_{neighbor1}$ quantifies the steric effects between n substituents on 1 and 4 and between substituents on 2 and 3 (see Figure 2.1); these have total size x_{14} and x_{23}. The $\rho_{neighbor2}$ part quantifies the steric effect between n substituents on 1 and 2 and between 3 and 4; these have total size x_{12} and x_{34}.

$$A_{steric}(\Delta H_f)_C = \rho_{neighbor1}(x_{14} + x_{23}) + \rho_{neighbor2}(x_{12} + x_{34}) \tag{2.6}$$

Acetylenic Steric The acetylenic reactophore also has one unique substituent location and is linear in geometry; the substituents cannot affect each other so only the total size (x_n) of the substituents is found and multiplied by the ρ_{steric} quantifier. (Equation 2.7).

$$A_{steric}(\Delta H_f)_C = \rho_{steric}x_n \tag{2.7}$$

Aromatic Steric The aromatic steric adjustment occurs in three parts. The regular steric adjustment occurs for all substituents, then if a substituent is ortho or perri to another substituent an additional correction must be made (Equation 2.8, see Figure 2.7).

$$A_{steric}(\Delta H_f)_C = A_{regular}(\Delta H_f)_C + A_{ortho}(\Delta H_f)_C + A_{perri}(\Delta H_f)_C \tag{2.8}$$

Aromatic Steric: Regular The $A_{regular}(\Delta H_f)_C$ is the adjustment to the aromatic reactophore for the all of the substituents, regardless of their location. ρ_{steric} quantifies what effect n substituents, with total size x_n, have on the aromatic reactophore.

$$A_{regular}(\Delta H_f)_C = \rho_{steric}x_n \tag{2.9}$$

Aromatic Steric: Ortho The ortho steric adjustment determines if an ortho substituent is next to one other substituent (ortho-out) or two substituents (ortho-in) and makes the appropriate correction (Figure 2.7). In equation 2.10, ρ_{outer} and ρ_{inner} quantify the effect substituents with total size x have on each other.

$$A_{ortho}(\Delta H_f)_C = \rho_{outer}x_{outer} + \rho_{inner}x_{inner} \tag{2.10}$$

Aromatic Steric: Perri The perri steric adjustment occurs if a substituent is perri to other substituents (Figure 2.7). Equation 2.11 expresses this as the sum of all the perri pairs on a reactophore, where ρ_{perri} quantifies the effect perri substituents have on each other; x_{perri} is the total size of these substituents.

$$A_{perri}(\Delta H_f)_C = \sum \rho_{perri}x_{perri} \tag{2.11}$$

Perturber Models: Resonance

The resonance models are fully described in references [2, 3, 4, 5]. Resonance is a perturbation affecting the reaction center through π-electron delocalization. As more charge is distributed out of the reaction center, the reaction center becomes stabilized; this stabilization is expressed as an overall lowering of energy in the molecule (Equation 2.12). Δq is the amount of charge distributed out of the reaction center and into the neighboring perturber groups and $\rho_{resonance}$ is the susceptibility of a given reaction center to this delocalization and quantifies the energy lowering.[2] This adjustment only occurs when there are π-electrons in the system: for ethylenic, aromatic, and acetylenic reaction centers.

$$A_{resonance}(\Delta H_f) = \rho_{resonance}(\Delta q)_C \tag{2.12}$$

Perturber Models: Connectivity

Connectivity in a molecule is defined as the bond energy change in the reaction center when a hydrogen atom is removed and replaced with another group. For all reactophores except aromatic ones, the connectivity perturbation is expressed as equation 2.13. In this equation, the value of $A_{connectivity}$ is simply dependant on the reactophore being perturbed ($\rho_{connectivity}$) and the number of non-hydrogen substituents (x).

$$A_{connectivity}(\Delta H_f)_C = \rho_{connectivity}(x) \tag{2.13}$$

The aromatic connectivity adjustment is unique because there are five distinct substituent locations possible in an aromatic reaction center (Figure 2.8). These locations are designated: *end*, *br-end*, *mid*, *outer*, and *inner*. The *end* location is simply on a terminating benzenoid structure, which is not bent; the *br-end* is also on a final straight benzenoid, however, it is next to a bridge atom; the *mid* location is between two bridge atoms; the *outer* location is on the outside of a bend; the *inner* location is on the inside of a bend and next to a bridge atom. The aromatic adjustment (Equation 2.14) is simply the number of substituents in a specific location (x_n) multiplied by the proper adjustment for that location. If a substituent is in n locations simultaneously, each location contributes $1/n^{th}$ of an adjustment to the total adjustment.

$$A_{connectivity}(\Delta H_f)_C = \rho_{end}x_{end} + \rho_{br-end}x_{br-end} + \rho_{in}x_{in} + \rho_{out}x_{out} + \rho_{mid}x_{mid} \tag{2.14}$$

2.3.2 Whole Molecule Models

The whole molecule can experience effects not attributable to any single reaction center or appended molecular structure. These effects include: adjustments due to the compound being a ring or other type of constrained structure, extra steric corrections due to nearest neighbor interactions, and additional resonance effects due to homoaromaticity. The whole molecule adjustment can be expressed as Equation 2.15.

$$(\Delta H_f)_{Whole\_molecule} = (\Delta H_f)_{rings} + (\Delta H_f)_{xres} + (\Delta H_f)_{xsteric} \tag{2.15}$$

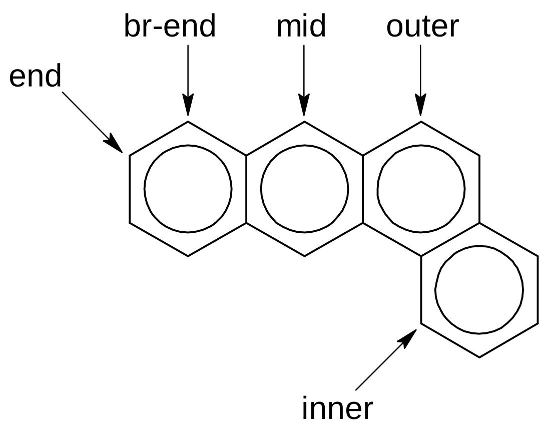

Figure 2.8: The five possible connectivity locations in an aromatic reaction center.

Whole Molecule Models: Rings

When atoms in a molecule are bonded in such a way to produce ring or cage structures, additional corrections are necessary to account for the extra strain. These corrections include: a basic correction for all the rings found in the system, having double or triple bonds as part of the ring, and additional structural corrections (Equation 2.16).

$$(\Delta H_f)_{rings} = \sum[\rho_{ring\_size}(n) + \rho_{ethylenic}(n, e) + \rho_{acetylenic}(n)] + \rho_{structure} \tag{2.16}$$

The ring size adjustment, $\rho_{ring\_size}(n)$, is simply based on the size of the ring being analyzed. Figure 2.9 is a graph of ring strain (the difference in the measured ΔH_f for cyclo-X-anes and the calculated ΔH_f without any ring strain correction) versus ring size. As ring size becomes smaller than six atoms, the ring strain increases; the same is true for rings larger than six until cyclodecane is reached, the ring strain then begins to decrease, approaching zero, with the exception of cyclododecane and cyclotetradecane. These two deviations could be due to poor measurement[8] or to some quirk of geometry that allows these compounds to have a lower ΔH_f than predicted. The quantifier, ρ_{ring_size}, is obtained as a functional fit of Figure 2.9's ring size data.

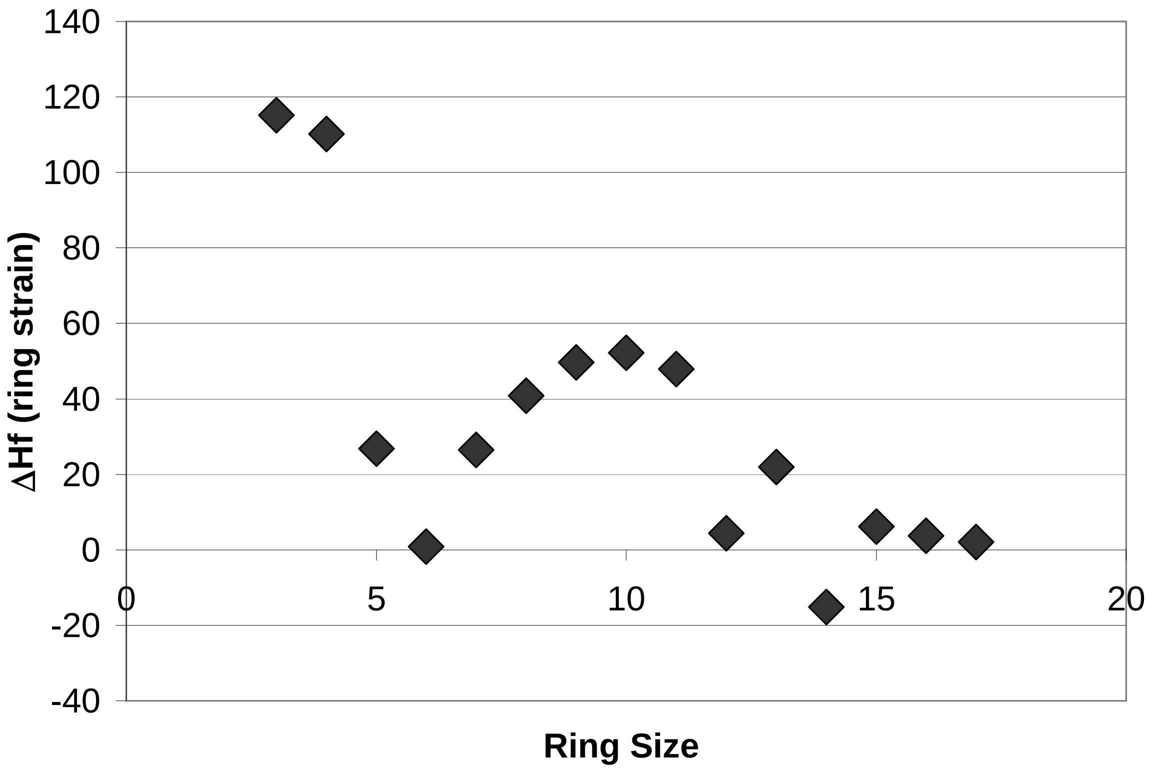

Figure 2.9: Ring strain versus ring size for cyclo-X-anes

The ethylenic adjustment, $\rho_{ethylenic}(n, e)$, is necessary when a double bond is endo or exo (e) in a ring; it is dependant on ring size (n), and is quantified by the multiplier $\rho_{ethylenic}$. If the ethylenic in question is in more than one ring and could be considered both endo and exo depending on which ring is being examined, only the endo correction is necessary. This is due to the major strain being in the endo ring and to avoid "over-correcting" the reaction center. If there are multiple double bonds in a ring, an additional size dependant correction must be made. If multiple double bonds are endo and connected, a resonance correction is subsumed into this correction.

An acetylenic adjustment, $\rho_{acetylenic}$, is necessary when an acetylenic reaction center is in a ring. $\rho_{acetylenic}$ quantifies the strain that results from bending the normally linear acetylenic substituents.

The ring structure adjustment, $\rho_{structure}$, consists of the additional structure adjustments: ρ_{side} and ρ_{cage}. The ρ_{side} adjustment corrects for when a ring shares a side with an aromatic reaction center, another ring, or with multiple rings. The amount of adjustment necessary is based on the size of the connected rings and on the location of the ring relative to the connected structure, i.e. is it connected along a side or at a single atom. When there are multiple fused rings in a molecule, it becomes very difficult to predict their ΔH_f from just local steric and connectivity models. Because of this difficulty, SPARC uses the ρ_{cage} adjustment to identify the type of structure present and quantify the amount of energy present in the structure. If multiple structures are recognized, a correction is applied for each structure found and the manner in which it is connected to other structures in the molecule. There are thirteen basic structures SPARC searches for identifies: bridge structures[9]; nortricyclene; adamantine; protoadamantane; tetrahedrane; cubane; bullvalene; Hexacyclo[5.4.1.02,6.03,10.05,9.08,11]dodecane; diadamantane; Dicyclopropa[cd,gh]pentalene,octahydro-; Tetracyclo[4.1.0.02,4.03,5]heptane; Prismane; and azulene. A sample of each of these structures is located in Figure 2.10.

Whole Molecule Models: Steric

The whole molecule steric adjustment is necessary to account for nearest neighbor steric interactions. This adjustment only occurs in methyl reaction centers, which are designated stereocenters, and is expressed as equation 2.17.

$$(\Delta H_f)_{xsteric} = \rho_{near\_neighbor}(m, n) \tag{2.17}$$

$\rho_{near\_neighbor}(m, n)$ quantifies the effect each pair of stereocenters (m,n) have on each other and on the entire molecule. This adjustment is similar to the *gauche* interaction described by Benson[10] and is used to describe the differences in the ΔH_f between molecules with different structural representations. For hydrocarbons, this difference is mainly found in ring systems (Figure 2.11). Rules have been developed which are dependant on the size of the ring the stereocenters are in; the size of the other ring the stereocenters are in, if both are

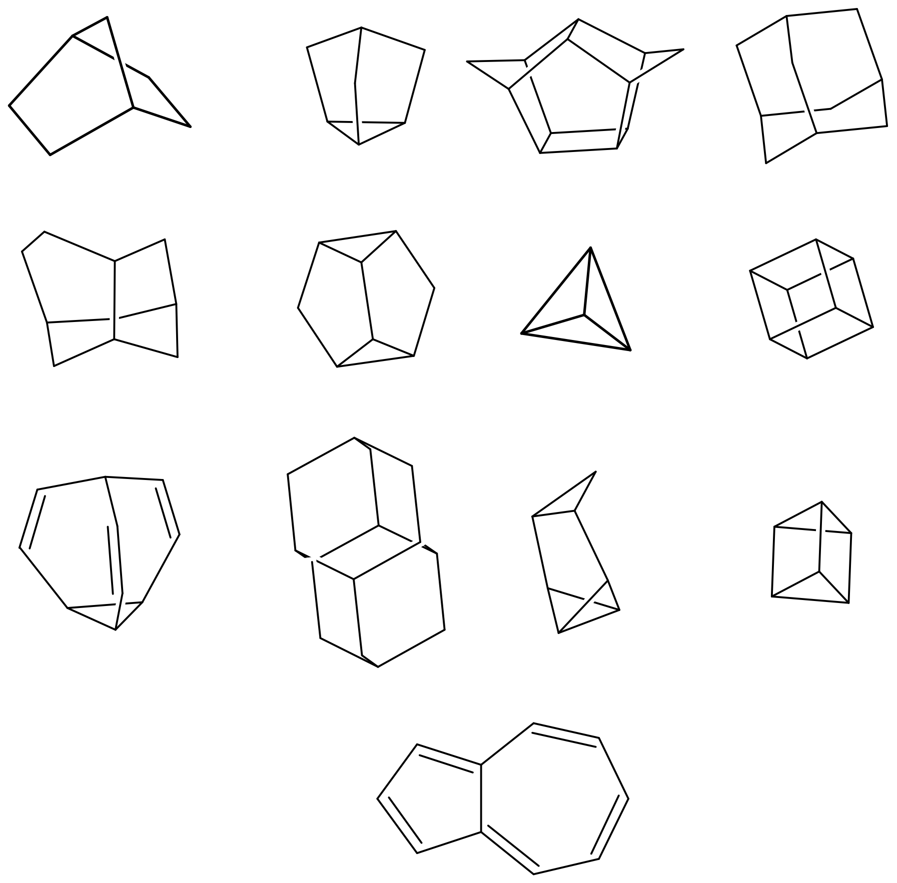

Figure 2.10: The various ring structures SPARC identifies (from left to right): bridge structures; nortricyclene; Hexacyclo[5.4.1.02,6.03,10.05,9.08,11]dodecane; adamantine; protoadamantane; Dicyclopropa[cd,gh]pentalene,octahydro-; tetrahedrane; cubane; bullvalene; diadamantane; Tetracyclo[4.1.0.02,4.03,5]heptane; Prismane; azulene

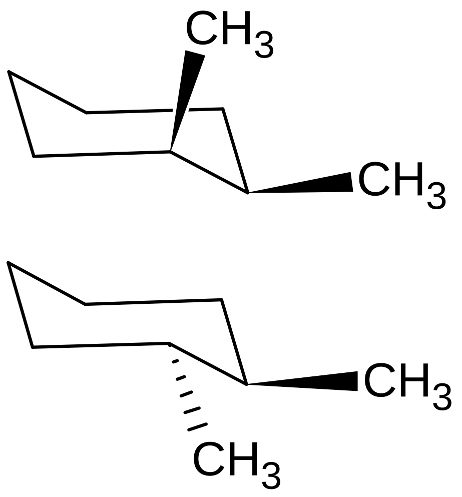

Figure 2.11: The difference in the confirmations of cis- and trans- dimethylcyclohexane.

in different rings; the position of the stereocenters in the ring(s), relative to each other; and the relative spatial orientation of each stereocenters substituents. After taking all of these rules into account, the correction for each interaction is approximately 5.5 kJ mol^{-1}. The only exception to this rule is for trans-Bicyclo[3.3.0]octane, which has a 25.2 kJ mol^{-1} increase in energy.

Whole Molecule Models: Resonance

An additional resonance correction is necessary when a molecule has homoaromaticity, such as triquinacene. This is expressed as equation 2.18.

$$(\Delta H_f)_{xres} = \rho_{harom}(l) \tag{2.18}$$

Homoaromaticty is a resonance correction that lowers the overall heat of formation because the structure of the molecule is of three rings joined in such a way that π-electron delocalization occurs through space and not across bonds.[11] This creates extra stability and an overall lowering of the enthalpy of formation. The stabilization for homoaromaticity is approximately 19 kJ mol^{-1}.

2.4 Results and Discussion

The SPARC method of calculating the heat of formation of hydrocarbons is based on structure query and analysis. It involves summing the value of each reaction center and its local perturbations and then analyzing the entire molecule to ensure other steric or resonant effects are taken into account. It is more successful in modeling this property than previous attempts using *ab initio*, semi-empirical, and group additivity methods.

There are multiple *ab initio* approaches to calculate the heat of formation. However, the results are nearly always the same: large basis sets lead to increased calculation time, and generally, but not necessarily, correspond to an increase in accuracy; however, as molecular size increases beyond 10 atoms the calculation time increases and the accuracy of the calculations rapidly deteriorates. For a typical small calculation (less than 10 atoms), the average error reported is 16 kJ mol^{-1} and the time to complete a calculation is measured in tens of hours.[12, 13]

There are many semi-empirical methods that have been employed in the calculation of ΔH_f. One of the most common and accurate methods, especially for organic compounds is PM3. PM3 was developed by Stewart as a reparametrization of the AM1 method specifically for organic molecules.[14] This method handles most compounds; however, the structures need to be optimized and even then theory occasionally breaks down, giving poor results. PM3 calculations are much quicker than *ab initio* calculations; however, this method still took over a week to calculate the ΔH_f of 574 compounds with an average RMS of 57.8 kJ mol^{-1}.[13, 15]

The group additivity method developed by Benson is both simple and accurate. Benson's approach is based on the premise that most properties of larger molecules can be considered as the sum of the contributions from the individual atoms or bonds in the molecule.[10] Benson extends this proposal from individual atoms to groups of atoms. Using the component groups of a molecule, the property in question can be calculated by summing the contributions from each group.[1, 7, 10] Benson derives the group data from the observed heats of formation of known compounds.[1, 7, 10] The Group Additivity method handles most compounds; however, if a group collection is not defined it will fail. There are a large number of parameters in this model[1, 7, 10], but it is extremely fast and accurate. An on-line calculator is available via the NIST chemical webbook[16] and was used to determine the ΔH_f of 447 compounds in less than 12 hours, with an RMS deviation of 17.7 kJ mol^{-1}.[13]

The SPARC method of calculating ΔH_f is similar to Benson's group additivity (GA) method. The major difference being that the GA method assigns every possible group situation a value, whereas the SPARC model infers the group constant by analyzing the local environment of the group. As a simple example, consider the series methane, ethane, propane, 2-methyl propane and neopentane. All of the carbons in these compounds start with a base (unperturbed) value of -74.87 kJ mol^{-1}, the value of the heat of formation of methane. The methane calculation finds this starting value and since there are no appended perturbing structures, this is the final answer. When calculating the ΔH_f contribution for groups with perturbing structures, a connectivity correction must be made. For methyl groups, this correction is expressed as: $C_n = -4.23n^2 + 35.53n$ where n is the number of non-hydrogen bonds to the carbon. For ethane, each carbon starts with the unperturbed -74.87 kJ mol^{-1} value and is perturbed for connectivity using the formula, giving a connectivity of 31.29 kJ mol^{-1} for each carbon. The next step would be to calculate the steric contribution of each group. The SPARC calculator sums the sizes (as described in reference [5]) of the n groups connected to the carbon and subtracts $n \times Methylsize$, where $Methylsize$ is the size of a single methyl group. This quantity is then multiplied by the steric susceptibility. For ethane, the steric contribution is zero since each carbon is connected to a methyl group. For propane the middle carbon would contribute -74.87 kJ mol^{-1} with a C_n of 54.12 kJ mol^{-1}. Since the perturbing structures connected to the central carbon are both of $Methylsize$ the steric correction is again zero. The end carbons are symmetry equivalent and each contributes a base of -74.87 kJ mol^{-1}, a connectivity of 31.29 kJ mol^{-1} and a steric contribution of $7.17 \times Stericdiff$, where $Stericdiff$ is the size of ethyl minus methyl. The steric contribution is calculated to be 0.29

Table 2.3: Individual components of the SPARC calculated series methane, ethane, propane, 2-methyl propane, and neopentane.

Name	**Base**	**Connectivity**	**Steric**	**Total**	**Observed**
Methane	-74.87	0.00	0.00	-74.87	-74.87
Ethane	-149.74	62.58	0.00	-87.16	-83.80
Propane	-224.61	116.70	0.58	-107.33	-105.64
2-methyl propane	-299.48	162.35	1.71	-135.42	-134.20
Neopentane	-374.35	199.53	3.96	-170.86	-167.56

kJ mol^{-1} for each end carbon, or 0.58 kJ mol^{-1} for the entire compound. For 2-methyl propane the three branched carbon contributes -74.87 kJ mol^{-1}, C_n is 68.48 kJ mol^{-1} and there is no steric correction, as all of its substituents are of $Methylsize$. The other three methyl groups are all equivalent and each has a steric contribution of $7.17 \times Stericdiff$, where in this case $Stericdiff$ is the difference in size between an isopropyl and methyl group. This leads to each having a -74.87 kJ mol^{-1} base value, a connectivity value of 31.29 kJ mol^{-1}, and a steric contribution of 0.57 kJ mol^{-1}. Finally, for neopentane, the central group contributes -74.87 kJ mol^{-1}, C_n is 74.37 kJ mol^{-1} and the steric contribution is 0 kJ mol^{-1} (*vide supra*). The four outer carbons are equivalent and each has a -74.87 kJ mol^{-1} base value, a C_n of 31.29 kJ mol^{-1}, and the steric contribution is $7.17 \times Stericdiff$, where $Stericdiff$ is the difference in size between a t-butyl group and methyl. This leads to a total steric contribution of 0.99 kJ mol^{-1}. These calculations are summarized in Table 2.3.

The experimental data used in this paper were obtained from NIST and literature sources.[8, 17, 18, 19] Figure 2.12 shows the calculated versus observed values for all 587 compounds. For these 587 compounds, the RMS deviation is 4.50 kJ mol^{-1} and the time to calculate is less than 30 seconds for the entire set.[13] Table 2.4 contains the RMS and R2 values of the SPARC, PM3, and Group Additivity methods. For SPARC, the RMS includes all 587 values, for PM3 it's for 574 compounds, and for the Group Additivity method the

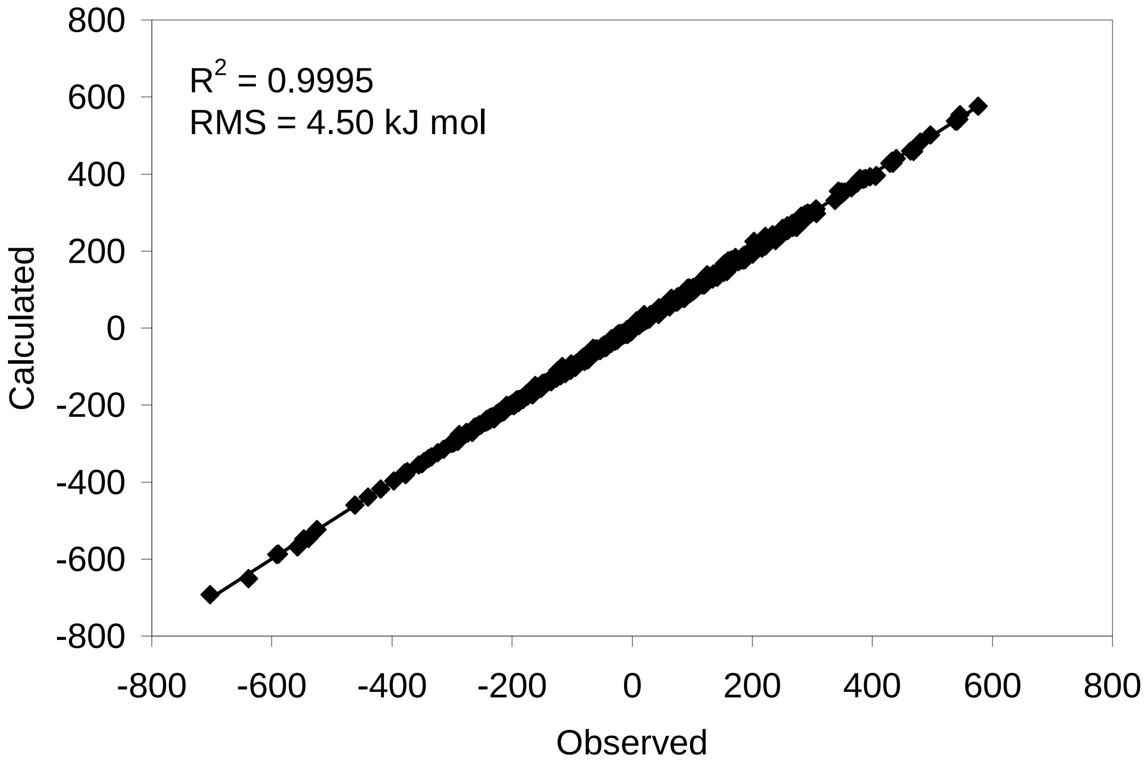

Figure 2.12: SPARC versus Observed for 587 heats of formation

RMS is for only 447 compounds.[13] The observed RMS value is included for the 587 measured compounds and is derived from the uncertainty reported in the measurement.

2.5 Conclusion

The extensive development of the hydrocarbon model is due to hydrocarbons having the most reliably measured heats of formation and also because it provides the basic structural framework for all other types of organic compounds. While at present SPARC only calculates the heat of formation of hydrocarbons, work is in progress to extend the capability of these models to calculate the ΔH_f for molecules containing nitrogen, oxygen, phosphorus, sulfur, and halogens. Once the heteroatom models are complete it will be possible to integrate them into SPARC's kinetic and redox rate models in order to better predict transition state

Table 2.4: Comparison of the error in observed values with the SPARC, PM3, and Group Additivity calculated RMS values. Also included is the R2 value.

	RMS	**R2**
Observed Values	2.95	
SPARC	4.50	0.9995
PM3	57.78	0.9292
Group Additivity	17.67	0.9911

energies. The extrapolatability of models to other types of chemistry is one of SPARC's greatest strengths and is the impetus behind continued development of these models.

2.6 References

[1] Cohen, N.; Benson, S. W.; *J. Chem. Rev.* 1993, 93, 2419-2438.

[2] Hilal, S. H.; Karickhoff, S. W.; Carreira, L. A.; *Quant. Struct.-Act. Relat.* 1995, 14, 348-355.

[3] Hilal, S. H.; Karickhoff, S. W.; Carreira, L. A.; *QSAR & Comb. Sci.* 2003, 22, 565-573.

[4] Hilal, S. H.; Karickhoff, S. W.; Carreira, L. A.; *Quant. Struct.-Act. Relat.* 1993, 12, 389-396.

[5] Hilal, S. H.; Carreira, L. A.; and Karickhoff, S. W. in *Theoretical and Computational Chemistry, Quantitative Treatment of Solute/Solvent Interactions*; Politzer, P.; Murray, J. S., Eds.; Elsevier Publishers: New York, 1994; pp 291-353.

[6] Carreira, L. A.; SPARC. http://ibmlc2.chem.uga.edu/sparc

[7] Stein, S. E.; Golden, D. M.; Benson, S. W. *J. Phys. Chem.* 1977, 81, 314-317.

[8] Cox, J. D. and Pilcher, G. *Thermochemistry of Organic and Organometallic Compounds*; Academic Press: London, 1970.

[9] Structures that intersect in at least three places and have only two atoms in common.

[10] Benson, S. W. *Thermochemical Kinetics*, 2nd ed.; John Wiley & Sons: New York, 1976.

[11] Liebman, J. F.; Paquette, L. A.; Peterson, J. R.; Rogers, D. W.; *J. Am. Chem. Soc.* 1986, 108, 8267-8268.

[12] Irikura, K.; Frurip, D.; *Computational Thermochemistry.* American Chemical Society: Washington DC, 1998.

[13] The number of compounds for each method varies for the following reasons. HyperChem "timed out" or otherwise generated an error for some compounds. The same is true for the NIST webbook. For NIST if a compound didn't work it was due to the group not being present, for HyperChem the compound was usually too complicated to analyze. We do not have results for ab inito calcuations because we do not have the facilities to run these compounds at this time.

[14] Stewart, J. J. P.; *J. Comp. Chem.* 1989, 10, 221-264.

[15] HyperChem(TM) Student Edition 7.0; Hypercube, Inc.; 1115 NW 4th St, Gainesville, Florida 32601.

[16] Stein, S. E.; Brown, R. L.; "Structures and Propeties Group Additivity Model" in NIST Chemistry WebBook, NIST Standard Reference Database Number 69, Linstrom, P. J.; Mallard, W. G., Eds; March 2003, National Institute of Standards and Technology: Gaithersburg MD, 20899 (http://webbook.nist.gov).

[17] Pedley, J. B.; Naylor, R. D.; Kirby, S. P. T*hermochemical Data of Organic Compounds*, 2nd ed.; Chapman and Hall: London, 1986.

[18] Roth, W. R.; Adamczak, O.; Breuckmann, R.; Lennartz, H. W.; Boese, R.; *Chem. Ber.* 1991, 124, 2499-2521.

[19] Afeefy, H. Y.; Liebman, J. F.; Stein, S. E.; “Neutral Thermochemical Data” in NIST Chemistry WebBook, NIST Standard Reference Database Number 69, Linstrom, P. J.; Mallard, W. G., Eds; March 2003, National Institute of Standards and Technology: Gaithersburg MD, 20899 (http://webbook.nist.gov).

Chapter 3

Prediction of the Enthalpy of Formation of Halogenated Hydrocarbons Using SPARC[1]

[1]Whiteside, T.S. and L.A. Carreira. Submitted to *J. Theoretical and Computational Chemistry*, 10/28/2004

3.1 Introduction

Halogenated hydrocarbons are carbon-based compounds where one or more hydrogen atoms have been replaced with a halogen. These compounds are widely used in industrial, commercial, and medicinal applications.[1] They are used in the final and intermediate products of manufacturing, in commercial applications as cooling agents and aerosol propellants, and medicinally as drug delivery agents and effective anesthetics.[1] Organohalogens are also found in the upper atmosphere and in local environments.[2] Because of their widespread use and existence, an accurate assessment of the physicochemical properties of these compounds is of great importance for both practical and theoretical uses.[1, 2]

The enthalpy of formation (heat of formation or ΔH_f) is one of the basic thermodynamic properties of molecules. Scientists use this property for many diverse applications; one practical application is in examining the reactivities of compounds to aid in the prediction of equilibrium constants.[3] We are interested in using the heat of formation model to improve the prediction of transition state energies. Improved transition state energies will be used in SPARC's reaction kinetics and reduction-oxidation models.

The heat of formation remains unknown for many halohydrocarbons; several methods, including *ab inito*, semi-empirical[4, 5], and group additivity[1, 3, 6, 7, 8, 9, 10], have been used to predict this basic quantity. All of these methods, with the exception of Joshi's[10], were developed primarily for "small" molecules; e.g. halomethanes and haloethanes.

SPARC (SPARC Performs Automated Reasoning in Chemistry) is a computer program developed to analyze molecular properties and reactions in the same manner as an expert chemist. SPARC uses the same basic method and general equations for each chemical property. This includes using SPARC's structural analysis engine to factor molecules into individual functional groups (reaction centers and perturbers) and then determining how each factored group affects the others in contributing to the overall property. The reaction centers, or reactophores, are the smallest subunits of the larger molecule with known chemical properties.[11] The basic quantity of each reaction center is a measured, unperturbed value.

Perturbers are molecular structures attached to a reactophore which affect its energy. If the reaction center is the only group in the molecule, there are no perturbing structures; hence, the value for the property is the value in SPARC's database of measured values. Otherwise, the reactophore's value will differ from the measured value by the sum of the perturbations. These perturbations are quantified into the various interactions commonly occurring in physical organic chemistry such as steric, resonance, connectivity, etc. Each reaction center is analyzed by a common method; however, the results of the analysis will differ depending on the reaction center, groups connected to the reaction center, and the property being determined. Once the value of each reactophore is determined, the entire molecule is analyzed to account for additional structural or resonance effects which impact the property of interest. Summing these values produces the SPARC calculated value for the property of interest. A complete description of the SPARC method can be found in references [11]-[14].

3.2 Theoretical Methodology

3.2.1 Calculation of ΔH_f Using SPARC

Reference [14] discusses the calculation of the heat of formation of hydrocarbons using SPARC. To calculate the ΔH_f of halohydrocarbons several adjustments were made to the model. Equation 3.1 describes how the ΔH_f is calculated from reaction centers and their perturber structures.

$$\Delta H_f = \sum_{C=1}^{n} [(\Delta H_f)_C + A_C] + (\Delta H_f)_{Whole\_molecule} \tag{3.1}$$

In this equation, $(\Delta H_f)_C$ is the intrinsic heat of formation of the reaction center, C, and is assumed to be unperturbed and independent of any perturbing structures. A_C describes the energy changes made to the reaction center by the perturber structure(s); the $(\Delta H_f)_{Whole\_molecule}$ represents additional energy due to the overall molecular structure.

Equation 3.2 shows how the reaction center perturbation, A_C, is further factored into the mechanistic components SPARC calculates for perturbations to the heat of formation of each reaction center.

$$A_C = A_{steric}(\Delta H_f)_C + A_{resonance}(\Delta H_f)_C + A_{connectivity}(\Delta H_f)_C \tag{3.2}$$

$A_{steric}(\Delta H_f)_C$, $A_{resonance}(\Delta H_f)_C$, $A_{connectivity}(\Delta H_f)_C$ describe how the size, resonance, and connectivity of the perturber(s) affect the energy of the reaction center. $A_{resonance}(\Delta H_f)_C$ quantifies the effects of electron delocalization by a perturber on the reaction center. $A_{connectivity}(\Delta H_f)_C$ is the change in a reactophore's energy when a hydrogen atom is replaced with another group. $A_{steric}(\Delta H_f)_C$ is the effect of a perturbers size on the energy of the reaction center. The connectivity and steric adjustments of reference [14] were modified to properly model halogenated hydrocarbons.

The "whole molecule" correction, $(\Delta H_f)_{Whole\_molecule}$, is the sum of corrections applied to the entire molecule; it includes corrections for rings, additional resonance, and additional steric energy (Equation 3.3). This model was not modified from that used in modeling hydrocarbons.[14]

$$(\Delta H_f)_{Whole\_molecule} = (\Delta H_f)_{rings} + (\Delta H_f)_{xres} + (\Delta H_f)_{xsteric} \tag{3.3}$$

3.2.2 Molecular Functional Groups

In the heat of formation model, all molecular functional groups are considered reaction centers and each group is analyzed to determine its contribution to the overall energy. In addition to the methyl, ethylenic, acetylenic, and aromatic reaction centers, the SPARC ΔH_f calculator has been extended to recognize the fluorine, chlorine, bromine, and iodine functional groups. The values of the halogen reaction centers are determined based on the type of reaction center (methyl, ethylenic, acetylenic, or aromatic) to which they are attached. The $(\Delta H_f)_C$ value of each halogen-carbon reaction center is the difference in energy of replacing a single hydrogen on a carbon-based reaction center with that halogen. The halogen reaction centers are regarded as static; perturbations only affect the carbon-based reaction center.

Halogenated Methyl Reaction Centers

Halogenated methyl reaction centers are carbon atoms connected to other atoms through sp3 bonds where at least one of these bonds is to a halogen. When connected to a methyl reaction center the $(\Delta H_f)_C$ values of fluorine, chlorine, bromine, and iodine are -204.6, -48.1, 5.6, and 58.8 kJ mol^{-1}, respectively.

Halogenated Ethylenic Reaction Centers

Halogenated ethylenic reaction centers are two carbon atoms connected to each other by a double bond and to four other atoms by single bonds where one of these must be a halogen. Fluorine, chlorine, bromine, and iodine have $(\Delta H_f)_C$ values of -197.4, -39.2, 16.7, and 66.4 kJ mol^{-1} when they are connected to an ethylenic reaction center.

Halogenated Acetylenic Reaction Centers

No accurate data is available for halogenated acetylenes. In order to have a complete model, the value assigned is based on the energy found in the other types of carbon-based reaction centers. The $(\Delta H_f)_C$ values for fluorine, chlorine, bromine, and iodine are set to -200, -40, 10, and 60 kJ mol^{-1}.

Halogenated Aromatic Reaction Centers

Aromatic reaction centers are those composed entirely of benzene subgroups and are more commonly known as poly-benzoic aromatic hydrocarbons (PBAHs).[14] When an aromatic reaction center is halogenated, a hydrogen is replaced with a halogen in one of five unique locations (Figure 3.1). The basic $(\Delta H_f)_C$ value is not affected by substituent location because this effect is modeled by $A_{connectivity}$, as explained in reference [14]. The $(\Delta H_f)_C$ values for fluorine, chlorine, bromine, and iodine, when connected to an aromatic reaction center, are -207.4, -38.3, 18.9, and 76.1 kJ mol^{-1}.

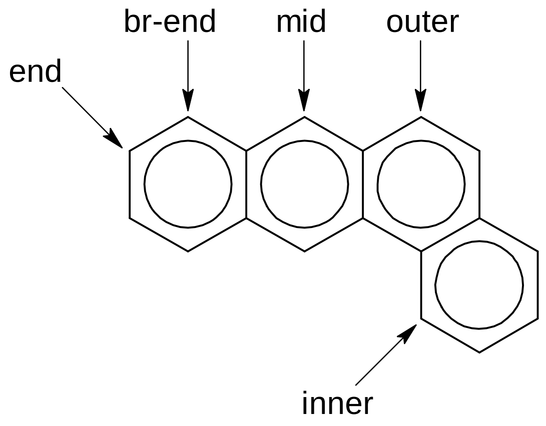

Figure 3.1: The five unique substituent locations in an aromatic reaction center.

3.2.3 ΔH_f Adjustment Models

Energy contributions are modeled in two separate processes: reaction center perturbations and whole molecule contributions. Reaction center perturbations are the sum of the effects of the perturber structures on a single reaction center; the whole molecule contribution models strain and resonance energy induced by the structure of the entire molecule. Only the reaction center perturbations have been modified to model halohydrocarbons; the whole molecule contributions are the same as for the hydrocarbon model.[14]

Reaction Center Models

Every reaction center has a basic value. With the exception of reactophores which may only bond to one other reaction center, this value is perturbed by any structure appended to the reaction center. These perturbations include steric, resonance, and connectivity. By extending the ΔH_f model to include halogens, only the connectivity and steric perturbation models were changed; the resonance perturbation remains as described in reference [14].

Perturber Models: Connectivity   Connectivity in a molecule is defined as the bond energy change in the reaction center when a hydrogen atom is removed and replaced with another group. The bond energy change is considered to be local to the reactophore in pure carbon-based reaction centers; substituents on neighboring reactophores do not affect it. For reaction centers with halogens, the effect of those other substituents on the bond energy of the reactophore must be modeled as well.[1, 6, 10]

Methyl Connectivity The original methyl connectivity model expresses $A_{connectivity}$ as Equation 3.4 with x being the number of non-hydrogen substituents.

$$A_{connectivity}(\Delta H_f)_C = \rho_{connectivity}(x) = ax^2 + bx + c \tag{3.4}$$

In the original hydrocarbon model, a and b are parameters of the methyl reaction center and c is 0 kJ mol^{-1}. With halogens, c is no longer zero; it represents the effects halogens have on the methyl reaction center and on each other (Equation 3.5).

$$c = Halo + Xi + Per \tag{3.5}$$

The $Halo$ interaction is the amount of energy contributed by multiple halogens bonded to a single atom (Figure 3.2). The energy of this interaction depends on the reaction center as well as the number and types of halogens bonded to it. In methyl reaction centers the $Halo$ interaction is composed of two functions (Equation 3.6).

$$Halo = Type1(F, Cl, Br, I) + Type2(Halo1, Halo2, Halo3, Halo4) \tag{3.6}$$

As the number of halogens of the same type are added to the methyl reaction center, the interaction of each pair changes quadratically. The $Type1$ function represents this interaction energy (Equation 3.7).

$$Type1 = \sum_{Type=F,Cl,Br,I} A_{Type}(NumType - 1)^2 + B_{Type}(NumType - 1) \tag{3.7}$$

The parameters A and B depend on the type of interacting halogens; $NumType$ is the number of halogens, of the specified $Type$, on the methyl reaction center. The $Type2$ function

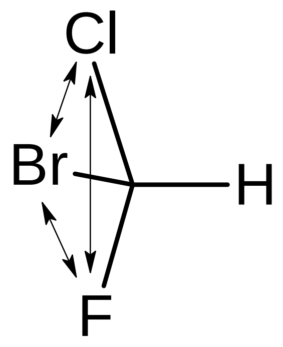

Figure 3.2: Halogenated methyl, showing the Br-F, Cl-F, and Br-Cl $Halo$ interactions.

is the sum of the interactions between the halogens on the methyl reaction center (Equation 3.8).

$$Type2 = (Halo1, Halo2) + (Halo1, Halo3) + (Halo1, Halo4) + \ldots \tag{3.8}$$

For the compound in Figure 3.2 the $Type1$ interaction is 0 kJ mol^{-1}, the $Halo$ contribution is entirely from the $Type2$ function. The three interactions are: Br-F (1.3 kJ mol^{-1}), Cl-F (4.2 kJ mol^{-1}), and Br-Cl (5.4 kJ mol^{-1}) or 10.9 kJ mol^{-1}. Table 3.1 contains a complete listing of parameters, interactions, and the energies associated with them.

The electronegativity of a halogen affects the bond strength between adjacent non-halogen atoms. The Xi correction quantifies the energy associated with this change and is dependant on the local structure of the molecule. There are two local structures which determine how the Xi interaction is calculated. One structure, $Struct1$, is a halogenated methyl reaction center which is bonded to a *non-halogenated* carbon-based reaction center. The other structure, $Struct2$, is a halogenated methyl reaction center which is bonded to a *halogenated* carbon-based reaction center. The Xi correction can be expressed as Equation 3.9.

$$Xi = Struct1(Halogens, Type) + Struct2(Halos1, Halos2) \tag{3.9}$$

The $Struct1$ correction is for reaction centers with the first type of structure and is simply a value determined by the interactions between halogens on the reaction center and the type

Table 3.1: The parameters, interactions, and energies associated with the Halo interaction, grouped by reaction center. Combinations not explicitly listed have a contribution of 0 kJ mol^{-1}

Reaction Center	**Parameter**	**Value (kJ mol^{-1})**
Methyl - Type1	A_F	-6.4
	B_F	-4.4
	A_{Cl}	7.5
	B_{Cl}	-5.1
	A_{Br}	0.4
	B_{Br}	12.4
	A_I	24.5
	B_I	-42.8
Methyl - Type2	br-cl	5.4
	cl-f	4.2
	br-f	1.3
	f-i	8.4
Ethylenic	f-f	-1.4
	cl-cl	2.3

of carbon-based reaction center to which the reaction center is bonded. In Figure 3.3, the reaction center 2 has this type of interaction with reaction center 3. The *Struct*1 correction is (Br,Cl-methyl) and has a value of 0 kJ mol^{-1}. The Struct2 correction is for reaction centers with the second type of structure and is determined by the interactions between each of the pairs of halogens on the reaction centers. In Figure 3.3, the reaction center 2 has this type of structure when paired with reaction center 1. The *Struct*2 correction is the sum of the (Br_2 ,Cl_1), (Br_2,F_1), (Cl_2,Cl_1), and (Cl_2,F_1) interacting pairs or 17.4 kJ mol^{-1}. This correction was developed solely for methyl-methyl pairs because there are no experimental data for methyl reaction centers with halogens bonded to other types of reaction centers with halogens on them. Table 3.2 lists the Xi interaction energies for which there is experimental data. If an interaction has not been measured its energy is assumed to be 0 kJ mol^{-1}.

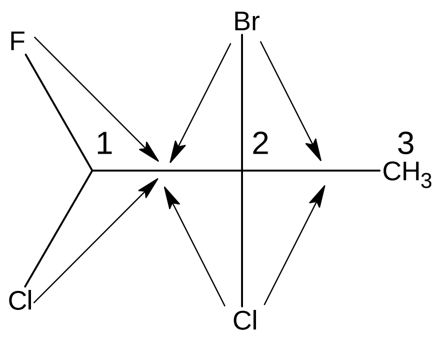

Figure 3.3: The Xi interaction of halogens on the bonds of neighboring atoms.

Table 3.2: The parameters and interaction energies, grouped by reaction center, of the Xi interaction for which there is experimental data. Combinations not explicitly listed have a contribution of 0 kJ mol^{-1}.

Reaction Center	**Parameter**	**Value (kJ mol^{-1})**
Methyl - Struct1	(f-methyl)	-16.7
	(cl-methyl)	-1.7
	(br-methyl)	-5.7
	(i-methyl)	-2.7
	(f,f-methyl)	-15.7
	(cl,cl-methyl)	-4.4
	(br,br-methyl)	-17.1
	(f,cl-methyl)	-13.5
	(cl,f,f-methyl)	27
	(br,f,f-methyl)	-15.4
	(i,f,f-methyl)	-2
	(f,f,f-methyl)	-24.3
	(cl,cl,cl-methyl)	5.6
	(f,f,f-ethylenic)	1.8
	(f,f,f-aromatic)	-16.5
Methyl - Struct2	f-f	-3.9
	cl-f	1.9
	br-f	-1.2
	f-i	-1.5
	cl-cl	1.6
	br-cl	15.1
Ethylenic - cis	f-f	24.7
	cl-cl	7.5
Ethylenic - trans	f-f	33

Table 3.3: The perfluorination parameters.

Reaction Center	**Type of Perfluoro Interaction**	**Value (kJ mol^{-1})**
Methyl	*per*1	-21.7
	*per*2	-0.2
	*per*3	36.2

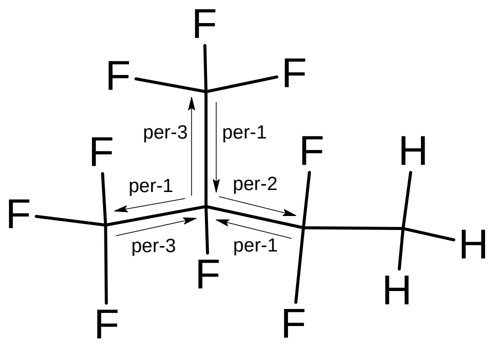

Figure 3.4: A perfluorinated alkane showing the Per parameter(s) used for interactions between reaction centers.

A perfluorinated compound is one in which all hydrogens have been substituted with fluorine. The *Per* parameter is non-zero when a fully fluorinated methyl reaction center is bonded to other fully fluorinated methyl reaction centers; the substituents on these other reaction centers must either be fluorine or carbon-based (no hydrogens). Perfluorinated compounds are unique because the ΔH_f is lower than would otherwise be predicted. The value of this parameter (Table 3.3) depends on the number of fluorines appended to the reaction center, which can be one, two, or three (Figure 3.4). For example, the total *Per* interaction for the reaction center with one fluorine in Figure 3.4 is the simply the sum of the *per*3, *per*3, and *per*2 parameters, or 72.2 kJ mol^{-1}.

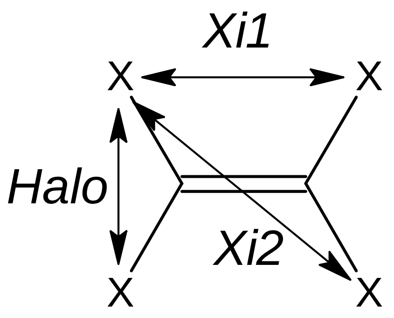

Figure 3.5: The interactions affecting one atom in an ethylenic reaction center.

Ethylenic Connectivity The original ethylenic connectivity model expresses $\rho_{connectivity}$ as equation 3.10; where a is a parameter for the ethylenic reaction center, x is the number of non-hydrogen substituents and c is 0.[14]

$$\rho_{connectivity}(x) = ax + c \tag{3.10}$$

With halogens, c represents the effects substituent halogens have on the reaction center as well as on each other. Equation 3.5 applies in this case also, with each subsection being determined slightly differently. $Halo$ is the interaction between halogens bonded to the same atom of the ethylenic reaction center. Xi is broken into two parts: $Xi1$ and $Xi2$ (Equation 3.11).

$$Xi = Xi1 + Xi2 \tag{3.11}$$

$Xi1$ is the *cis* interaction and $Xi2$ is the *trans* interaction. Figure 3.5 shows the Xi and $Halo$ interactions in an ethylenic reaction center. There are only two measured perfluorinated ethylenics and the SPARC calculated heat of formation is in good agreement with the measured values for these compounds. For these reasons no Per correction was necessary for these types of molecules.

Acetylenic Connectivity In halogenated acetylenics, equation 3.10 also applies, a uses the same values for halogens as for methyl substituents and c is 0.[14] This is because no accurate measurements have been made on halogenated acetylenics and the connectivity adjustments are subsumed into the reaction center correction.

Aromatic Connectivity No modification to the connectivity adjustment[14] of aromatic reaction centers is necessary; primarily because there are no halogenated aromatics larger than naphthalene with measured data.

Perturber Models: Steric Steric perturbations can play a major role in affecting the energy of reaction centers. The function of this interaction is dependant on the reaction center being perturbed as well as the size and type of appended structures.[14] With the addition of halogens to the model, the steric perturbation can also be described as an electrostatic repulsion.

Aromatic Steric All reaction centers, except the aromatic reaction center, subsume the steric effect into the $Halo$ and Xi portions of the connectivity corrections. In aromatic reaction centers every substituent is on a unique atom and steric interactions will occur between substituents which are ortho to each other. There are two types of ortho substituents: "inner" and "outer". An "outer" substituent is next to only one other substituent; an "inner" substituent is between two substituents (Figure 3.6).[14] The "ortho" correction, Equation 3.12, describes how ortho substituents 1 through n affect the energy of an aromatic reaction center through electrostatic repulsion. The $OType$, *outer* or *inner*, of each substituent determines the correction for that substituent.

$$A_{ortho}(\Delta H_f)_C = \sum_{x=1}^{n} \rho_{OType}(x) \tag{3.12}$$

ρ_{outer} and ρ_{inner} quantify the electrostatic repulsion between ortho substituents and are fully expressed in a generalized form as equation 3.13.

$$\rho_{OType}(x) = A_{OType} \times Type(x) + B \times TotalSize(x) \tag{3.13}$$

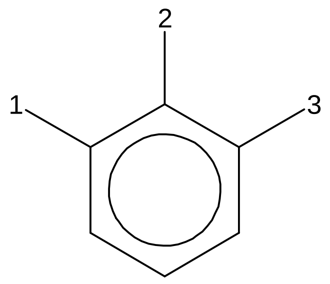

Figure 3.6: An aromatic reaction center where 1 and 3 are "outer" substituents and 2 is an "inner" substituent.

A_{Otype} is a multiplier which depends on whether the substituent is "outer" or "inner". The value the $Type(x)$ function returns is dependant on the both the type of ortho substituent ("outer" or "inner") and on the reactophore type of the interacting substituents (methyl, aromatic, fluorine, etc). If x is the "outer" substituent, 1 (Fig 3.6), $Type(x)$ is expressed as equation 3.14; if x is the "inner" substituent, 2 (Fig 3.6), $Type(x)$ is expressed as equation 3.15.

$$Type(1) = outer(Sub1, Sub2) \tag{3.14}$$

$$Type(2) = inner(Sub1, Sub2) + inner(Sub2, Sub3) \tag{3.15}$$

The B parameter is constant for both types of ortho substituents. $TotalSize(x)$ is determined as described in references [11] and [14] and is the total size of the substituents. For the "outer" substituent, 1 (Fig 3.6), $TotalSize(x)$ is expressed as Equation 3.16. The size function of "inner" substituent, 2 (Fig 3.6), is expressed as Equation 3.17.

$$TotalSize(1) = Size(1) + Size(2) \tag{3.16}$$

$$TotalSize(2) = Size(1) + Size(2) + Size(3) \tag{3.17}$$

The values of the individual parameters are listed in Table 3.4.

Table 3.4: Aromatic reaction center steric parameters and values. If an interaction is not defined, a multiplier of 1 is used instead.

Parameter	**Type**	**Value (kJ mol^{-1})**
A	Outer	-1.5
	Inner	2.5
B	Inner_outer_size	66.8
Type(x)	Outer-(cl,cl)	0.7
	Outer-(f,f)	-8.8
	Outer-(aromatic,f)	8
	Inner-(cl,cl)	2.5
	Inner-(f,f)	8.9
	Inner-(f,methyl)	-3
	Inner-(cl,methyl)	-1.3
	Inner-(f,i)	15.1
	Inner-(methyl,f3)	35.9
	inner-(aromatic,f)	26

3.3 Results and Discussion

Several methods have been developed to accurately predict the ΔH_f for halogenated hydrocarbons.[1, 3, 6, 9, 10] Unfortunately, most of these apply to a limited number of systems and are highly parameterized. Joshi's[10] generalized, bond-energy, group additivity method works as well as the SPARC method but is complicated and not computerized.

The SPARC method of calculating the heat of formation is based on structure query and analysis. It involves summing the value of each reaction center and its local perturbations and then analyzing the entire molecule to ensure other global effects are taken into account. It is more successful in modeling this property than either the semi-empirical PM3 method or Benson's group additivity method.

Many semi-empirical methods have been employed to calculate the ΔH_f of halogenated hydrocarbons.[4] One common method to calculate the ΔH_f is PM3. This method handles most compounds; however, the structures need to be optimized and even then theory occasionally breaks down, giving poor results. PM3 calculations are quick with respect to *ab initio* calculations; however, this method still took several days to calculate the ΔH_f of 196 compounds. The RMS of these halohydrocarbons was 41.8 kJ mol^{-1}.[15, 16]

The group additivity method developed by Benson is simple and accurate for a large number of molecules. Benson's approach is based on the premise that properties of molecules can be considered the sum of the contributions from the individual atoms or bonds in the molecule.[6] Benson derives the group data from the observed heats of formation of known compounds.[3, 6, 7] The group additivity method handles most compounds; however, if a group collection is not defined it will fail. An on-line calculator is available via the NIST chemical webbook[8] and was used to determine the ΔH_f of 169 compounds in less than 6 hours, with an RMS deviation of 39.1 kJ mol^{-1}.[15] The SPARC method of calculating ΔH_f is similar to Benson's group additivity method. The major difference being that the Benson method assigns every possible group situation a value, whereas the SPARC model infers the group constant by analyzing the local environment of the group. Figure 3.7 shows the SPARC calculated versus the observed values for all 202 compounds. SPARC calculates these 202 compounds with an RMS deviation of 5.18 kJ mol^{-1} in less than 30 seconds.

The experimental data used in this paper were obtained from NIST and literature sources.[1, 4, 9, 17, 18, 19] To ensure quality data, if data measured before 1985 were not included in references [17] and [18], it was discarded. The observed RMS value is derived from the uncertainty reported in the measurement and is 5.61 kJ mol^{-1} for these 202 compounds.

3.4 Conclusion

The SPARC ΔH_f model is nearly as accurate as most experimental measurements for hydrocarbons and halogenated hydrocarbons. It has been tested on all reliably measured halohydrocarbons. The structures of these compounds range from highly branched perfluoroalkanes to halogenated aromatics. SPARC provides a quick, reliable method for determining the ΔH_f of these compounds and an alternative to using other semi-empirical or group additivity

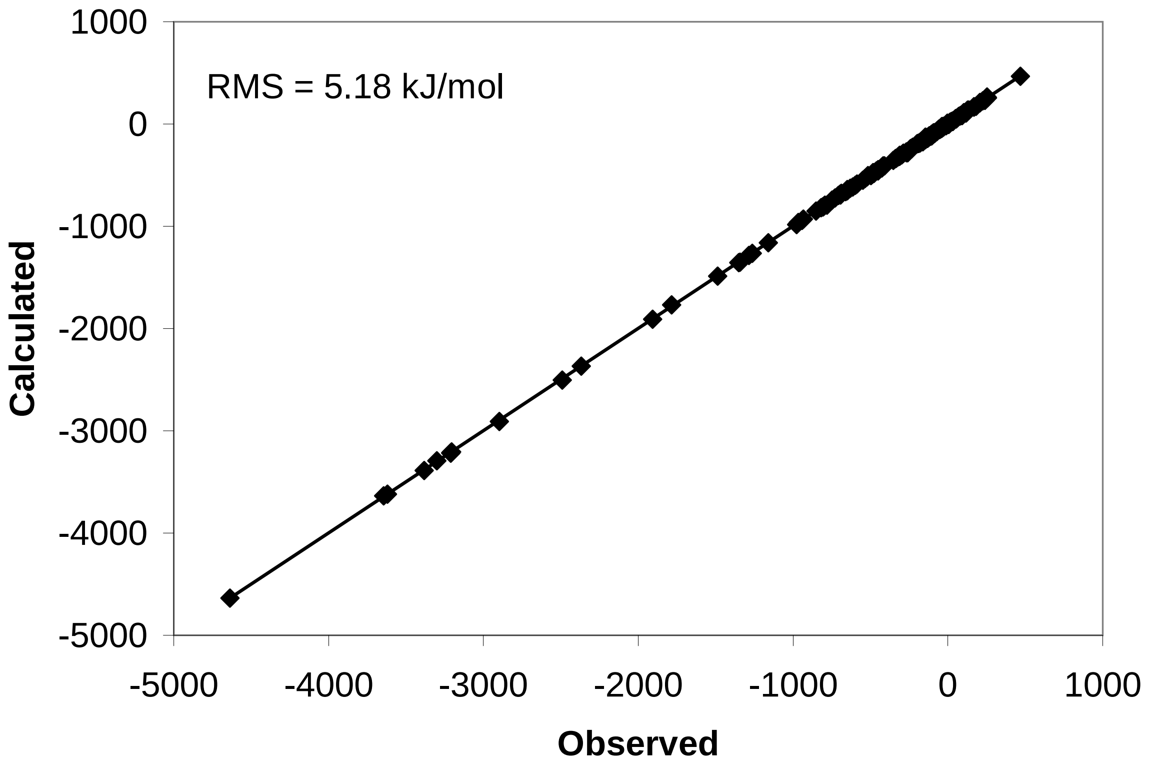

Figure 3.7: SPARV vs Observed for 202 heats of formation of halohydrocarbons

methods. While at present SPARC only calculates the heat of formation of hydrocarbons and halohydrocarbons, work is in progress to extend the capability of these models to calculate the ΔH_f for molecules containing nitrogen, oxygen, phosphorus, and sulfur. Once the heteroatom models are complete it will be possible to integrate them into SPARC models to better predict transition state energies. The extrapolatability of models to other types of chemistry is one of SPARC's greatest strengths and is the impetus behind continued development of this method. A complete listing of the compounds used to develop this model can be found in Appendix B.

3.5 References

[1] Kolesov, V. P.; Papina, T. S.; *Russian Chemical Reviews* 1983, 52, 425-439.

[2] Paddison, S. J.; Tschuikow-Roux, E.; *Int. J. Thermophysics* 1998, 19, 719-730.

[3] Cohen, N.; Benson, S. W.; *J. Chem. Rev.* 1993, 93, 2419-2438.

[4] Smart, B. E. in *Molecular Structure and Energetics* Vol 3.; Liebman, J. F.; Greenberg, A., Eds.; VCH Publishers: Deerfield Beach, FL, 1986; pp 141-191.

[5] Stewart, J. J. P.; *J. Comp. Chem.* 1989, 10, 221-264.

[6] Benson, S. W. *Thermochemical Kinetics*, 2nd ed.; John Wiley & Sons: New York, 1976.

[7] Stein, S. E.; Golden, D. M.; Benson, S. W. *J. Phys. Chem.* 1977, 81, 314-317.

[8] Stein, S. E.; Brown, R. L.; "Structures and Propeties Group Additivity Model" in NIST Chemistry WebBook, NIST Standard Reference Database Number 69, Linstrom, P. J.; Mallard, W. G., Eds; March 2003, National Institute of Standards and Technology: Gaithersburg MD, 20899 (http://webbook.nist.gov).

[9] Kolesov, V. P.; Kozina, M. P. *Russian Chemical Reviews* 1986, 55, 912-928.

[10] Joshi, R. M. *J Macromol Sci-Chem* 1974, A8, 861-885

[11] Hilal, S. H.; Carreira, L. A.; Karickhoff, S. W. in *Theoretical and Computational Chemistry*, Quantitative Treatment of Solute/Solvent Interactions; Politzer, P.; Murray, J. S., Eds.; Elsevier Publishers: New York, 1994; pp 291-353.

[12] Hilal, S. H.; Karickhoff, S. W.; Carreira, L. A.; *Quant. Struct.-Act. Relat.* 1995, 14, 348-355.

[13] Hilal, S. H.; Karickhoff, S. W.; Carreira, L. A.; *QSAR & Comb. Sci.* 2003, 22, 565-573.

[14] Whiteside, T. S.; Carreira, L. A.; *J. Theoretical and Computational Chemistry* 2004, 3, 451-469.

[15] The number of compounds for each method varies for the following reasons. HyperChem "timed out" or otherwise generated an error for some compounds. The same is true for the NIST webbook. For NIST, if a compound didn't work it was due to the group not being present, for HyperChem the compound was usually too complicated to analyze.

[16] HyperChem(TM) Student Edition 7.0; Hypercube, Inc.; 1115 NW 4th St, Gainesville, Florida 32601.

[17] Cox, J. D. and Pilcher, G. *Thermochemistry of Organic and Organometallic Compounds*; Academic Press: London, 1970.

[18] Pedley, J. B.; Naylor, R. D.; Kirby, S. P. *Thermochemical Data of Organic Compounds*, 2nd ed.; Chapman and Hall: London, 1986.

[19] Afeefy, H. Y.; Liebman, J. F.; Stein, S. E.; "Neutral Thermochemical Data" in NIST Chemistry WebBook, NIST Standard Reference Database Number 69, Linstrom, P. J.; Mallard, W. G., Eds; March 2003, National Institute of Standards and Technology: Gaithersburg MD, 20899 (http://webbook.nist.gov).

Chapter 4

Stereochemistry and the Enthalpy of Formation[1]

[1]Whiteside, T.S. and L.A. Carreira. To be submitted to *J. Theoretical and Computational Chemistry*

4.1 Introduction

SPARC (SPARC Performs Automated Reasoning in Chemistry) is a computer program developed to analyze molecular properties and reactions. SPARC uses a structural analysis engine to factor molecules into individual functional groups. It then determines how each factored group affects the other groups in contributing to the overall property. A complete description of the SPARC method is provided in references [2, 3, 4, 5, 6, 7].

The enthalpy of formation (heat of formation or ΔH_f) is one of the basic thermodynamic properties of molecules and finds uses in many applications. We are interested in using heat of formation models to improve the prediction of transition state energies. Improved transition state energies will aid in the development of reaction kinetics and reduction-oxidation models.

The heat of formation is a structurally dependant property. For many compounds it can be successfully modeled using the interactions of neighboring groups. This method incompletely describes the interactions occuring in some compounds and additional modeling is required for these interacting, non-neighboring groups. Benson recognized this non-neighboring interaction occurs between chiral centers and called it the *gauche* interaction.[1, 8, 9] He found it raises the ΔH_f by approximately 3.3 kJ mol^{-1} per interaction.

It was our goal to model the effect of stereochemistry on the heat of formation, of any organic compound. We examined the difference in the heat of formation between stereoisomers to quantify the effects different spatial environments have on the strain energy of each chiral center. Modeling the strain as a function of environment allows a more specific and robust model to be constructed. The advantage of this model can be clearly seen when compared to PM3 and NIST/Benson calculations. SPARC calculates a ΔH_f for all of the compounds used to build this model and is able to differentiate between 58 of the 70 sets of stereoisomers. The same set of molecules were calculated using PM3[10, 11] and the NIST calculator[12]; however, PM3 failed to calculate a ΔH_f for 3 of the compounds and the NIST calculator could only calculate 35 of the 169 molecules in the model. PM3 could only differentiate between 33 of the 70 stereoisomer sets and the NIST calculator, a group additivity calculator using Benson's rules, could only differentiate between 2 of the 70 sets of stereoisomers.

4.2 Theoretical Methodology

4.2.1 Calculation of the ΔH_f Using SPARC

The calculation of the heat of formation using SPARC has been rigorously discussed in references [6] and [7]. Equation 4.1 describes how the ΔH_f is calculated for a molecule.

$$\Delta H_f = \sum_{C=1}^{n} [(\Delta H_f)_C + A_C] + (\Delta H_f)_{Whole\_molecule} \tag{4.1}$$

In this equation, $(\Delta H_f)_C$ is the intrinsic heat of formation of the reaction center, C, and is assumed to be unperturbed and independent of any perturbing structures. A_C describes adjustments made to the reaction center by appended perturber structure(s); the $(\Delta H_f)_{Whole\_molecule}$ is the additional energy due to the overall molecular structure.

The "whole molecule" correction, $(\Delta H_f)_{Whole\_molecule}$, is the sum of corrections applied to the entire molecule; this includes corrections for rings, additional resonance, and additional steric energy (Equation 4.2).[6] $(\Delta H)_{xsteric}$ models the strain induced by various spatial environments. A simplified version, to model hydrocarbons, was described in reference [6].

$$(\Delta H_f)_{Whole\_molecule} = (\Delta H_f)_{rings} + (\Delta H_f)_{xres} + (\Delta H_f)_{xsteric} \tag{4.2}$$

4.2.2 Whole Molecule Models: xsteric

To model the effects of stereochemistry, all chiral centers in the molecule are identified as one of two types of stereocenters. The first type is a linear chiral center; it is any chiral center which is not a ring atom. The second type is a chiral center which is also a ring atom. This second type of stereocenter is further sorted by the type of ring-structure in which it is found. These structures are:

1. Single rings (Fig. 4.1a)

2. Sections of a molecule containing a bridge (Fig. 4.1b)

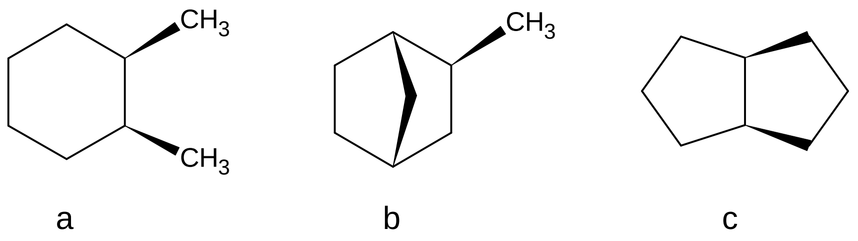

Figure 4.1: Three ring environments with stereochemistry: a. cis-dimethyl-cyclohexane b. exo-2-methyl-Bicyclo[2.2.1]heptane c. cis-octahydro-Pentalene

3. Sections of a molecule containing a ring sharing a side with another ring (Fig. 4.1c)

The model determines the strain energy in each chiral center and sums it to get the total steric energy (Equation 3).

$$(\Delta H_f)_{xsteric} = Linear + Single + Bridge + SideShare \tag{4.3}$$

Liner Chiral Centers

In a linear chiral center, the senses are defined but can not be used to specify spatial orientation because the molecule will rotate to relieve as much steric strain between substituents as possible. Because the chain rotates, SPARC is unable to distinguish between the spatial orientation of each substituent, therefore the *Linear* energy is set to 0 kJ mol^{-1}.

Single Ring Chiral Centers

In single rings, there are two energy contributions involving chiral centers. The energy of chiral centers in a single ring is expressed using Equation 4.4.

$$SingleRing = Steric + Axial \tag{4.4}$$

The *steric* interaction is the non-neighboring steric interaction, as described by Benson.[8] The *axial* function models the rotational strain energy of substituents in the axial position.

To model chiral centers in a ring, information about the ring's overall structure is necessary. This includes the size, type, and position of the substituents found on each chiral center. Each of the ring-atom's substituents has a *sense*; the *sense* defines each substituent's spatial position, relative to the ring. If a ring is observed edge-on, there is a top half and a bottom half. In order to determine the axial and equatorial positions of the substituents, the largest substituent is considered to be connected to the "first" ring-atom. The *sense* of this substituent is considered to be in the 1-direction and equatorial; the other substituents' *senses* are then set relative to this one. If a ring atom has a substituent in one half of the ring, its *sense* is 1; if this substituent is in the other half, its *sense* is 2. When there are two substituents off of the "first" ring atom, the largest is in the 1-direction and the other is in the 2-direction. If both substituents are the same size, their *senses* are not changed.

A *steric* interaction occurs between chiral centers bonded to each other, where both of these are in a ring and where substituents of these chiral centers have the same sense. Six-membered rings have a *Steric* interaction energy of 0 kJ mol^{-1} because the chair conformation of the ring prevents these substituents from interacting. The *Steric* interaction is the sum of the interactions between each pair of bonded chiral centers and depends on the size and type of interacting substituents (Equation 4.5).

$$Steric = \sum Size \times Type \times x \tag{4.5}$$

Size is the total normalized size of the substituents in the interacting pair, *Type* is a multiplier based on the substituents off of the chiral centers, and x is the chiral interaction energy. The chiral interaction energy, x, is 5.5 kJ mol^{-1} and quantifies the *Size* and *Type* multipliers into an interaction energy. The size of each substituent in the interacting pair, using previous SPARC models[3, 6], has been normalized versus the size of a methyl substituent (Equation 4.6 to determine the *Size* multiplier.

$$Size = \frac{size(Sub1)}{size(Methyl)} + \frac{size(Sub2)}{size(Methyl)} \tag{4.6}$$

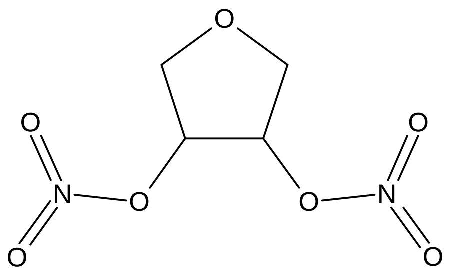

Figure 4.2: Oxolane-3,4-diol dinitrate

Type is a special multiplier developed because of the ΔH_f of cis- and trans- isomers of Oxolane-3,4-diol dinitrate (Fig. 4.2), as measured by Eremenko, Korolev, et al.[13] Unlike other single-ring compounds, the "steric" interaction occurs in the trans-isomer of this molecule; also, this "non-interaction" is four times larger than the interaction in other systems. The *Type* multiplier for these systems is 4, for all other systems it is 1.

Eremenko, et al. mention several possibilities as to why this anomaly could occur, including: conformational energies, interaction of substituents, and the crystal packing characteristics of these molecules. The first two possibilities do not occur in any other single-ring compound and, as explained below, the crystal packing characteristics also are unlikely to cause this difference.

According to Eremenko, the cis-isomer was a solid for the combustion measurement, but the trans-isomer was a liquid.[13] Additionally, the cis-isomer's boiling point is reported to be only 1C higher than the trans-isomer. Since the boiling points are so similar, each isomer's enthalpy of vaporization should also be similar. An analysis of similar compounds shows that the crystal packing should not contribute more than 5 kJ mol^{-1}. For these two reasons, the only conclusion is either some force besides the crystal packing characteristics causes the cis-isomer to be 20 kJ mol^{-1} more stable than the trans-isomer or the reported values should be reevaluated.

The *axial* energy term occurs only in four- and six-membered rings. These rings are "puckered", so substituents of these ring atoms are in either axial or equatorial positions. The axial position is energetically unfavorable; substituents in this position want to rotate into an equatorial position. The model analyzes the sense of each substituent and determines which are axial. The amount of strain energy due to each axial substituent depends on its size, $Size$, and the axial energy, y (Equation 4.7).

$$Axial = \sum Size \times y \tag{4.7}$$

$Size$ is determined using the same method as for $Steric$ chiral centers. The axial energy, y, quantifies the strain of a substituent on the ring; this energy is 5.5 kJ mol^{-1}.

Bridge Section Chiral Centers

Calculating the energy of chiral centers in a bridge section of a molecule is a multi-step process. First, the two smallest rings composing a bridge structure[6] are removed, leaving the largest of the three. The relative sense of each substituent of this ring is determined and set in the same manner as for a single ring. The chiral centers in the ring are classified as ChiralBridgeAtoms (CBA) or as OtherCAs. A CBA is a bridge atom; an OtherCA is any chiral center in the ring that is not a CBA. OtherCAs are found in three local environments. Environment 1 is an chiral atom sharing a side between two bridge sections in a molecule, Fig. 4.3, OtherCA1; Environment 2 is a chiral center with a substituent appended to it, Fig. 4.3, OtherCA2; and Environment 3 is a chiral center sharing a side between a bridge section and a single ring, Fig. 4.3, OtherCA3.

The overall energy for chiral centers in a bridge molecule is obtained by summing the energy of the OtherCAs in each environment (Equation 4.8).

$$Bridge = Env1 + Env2 + Env3 \tag{4.8}$$

The CBAs do not contribute to the strain energy because their energy is already accounted for in the $(\Delta H_f)_{rings}$ adjustment.[6]

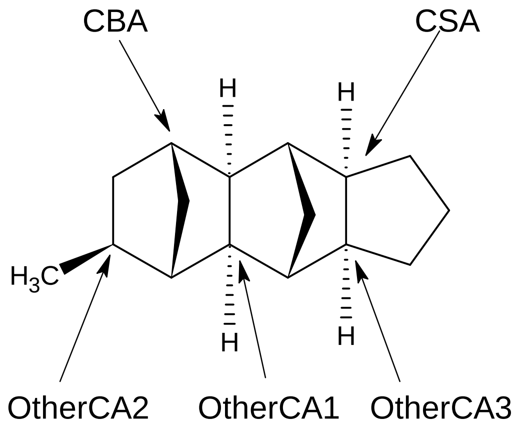

Figure 4.3: Two bridge sections showing the three local environments of OtherCAs. Also shown is a ChiralBridgeAtom (CBA) and a ChiralSideShareAtom (CSA).

The energy of each OtherCA in Environment 1 is determined by the senses of each set of CBA-OtherCA-CBA. These sets have three unique combinations of senses: x-x-x, y-x-y, or x-y-y (Figure 4.4). These three combinations describe all possible interactions between adjacent bridge sections. If each member of the set has the same sense, x-x-x, there is one interaction; if both CBAs have the same sense and the OtherCA has the opposite sense, y-x-y, there are two interactions; if one CBA and the OtherCA have the same sense and the other CBA has the opposite sense, x-y-y, there are zero interactions. The energy for each OtherCA in Environment 1 is the number of interactions, NI, quantified by the chiral interaction energy, x (Equation 4.9).

$$Env1 = \sum NI \times x \tag{4.9}$$

For an OtherCA in Environment 2, the energy is determined using a similar method as for a chiral center in a single ring. The difference in the two methods is that strain is placed on the chiral center when the CBA and the OtherCA have opposite senses, as opposed to the same sense. This is because the ring is not planar and when the senses are opposite, the substituent is oriented similar to an axial position, with the same type of strain. $Env2$ is a

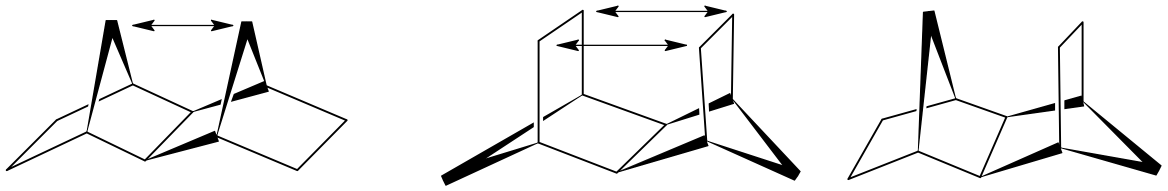
Figure 4.4: The various interactions which occur between the three unique bridge environments (x,x,x; y,x,y; x,x,y).

function of the size of the substituent, *Size*, and the axial interaction energy, y (Equation 4.10).

$$Env2 = \sum Size \times y \tag{4.10}$$

The amount of strain on OtherCAs in Environment 3 is determined by the size and sense of the ring. Like Environment 2, no strain is placed on the OtherCAs except when the substituents and neighboring CBAs have opposing senses. Equation 4.11 is used to model the strain energy on an OtherCA in Environment 3.

$$Env3 = \sum Ring \times y \tag{4.11}$$

If there are more than four ring-atoms, *Ring* is 1; for smaller rings, *Ring* is 2.5. This reflects the additional strain in the system due to the smaller bond angles. The strain on each OtherCA is quantified by *Ring* and the axial interaction energy, y.

SideShare Chiral Centers

To calculate the energy of chiral centers connecting two rings, each pair of connecting rings must be examined. First, the chiral centers are sorted into Chiral SideShare Atoms (CSAs) and OtherCAs. Chiral SideShare Atoms (Figure 4.3 and Figure 4.1c) are the chiral centers which join two rings. If one of the rings is part of a bridge structure this correction does not occur since it is already modeled in the $Env3$ function of a bridge section. The OtherCAs are corrected based on their local environment, as described in the previous sections. Each

Table 4.1: The SS multiplier for CSA pairs with rings of various senses and lengths. For all other combinations no correction is made.

Sense	**Length Ring1**	**Length Ring2**	**SS**
Cis	< 5	> 7	1
Trans	< 5	< 8	1
Trans	5	6	1
Trans	5	5	4.72
Cis	6	6	3.27

CSA is examined to determine both its sense and the other CSA to which it is bonded. The size of each ring on either side of the CSA pair is also determined. Once this information is known, the $SideShare$ correction can be made (Equation 4.12).

$$SideShare = \sum SS(sense, ring1, ring2) \times x + Middle \tag{4.12}$$

The total strain energy of a CSA pair is determined by the function $SS(sense, ring1, ring2)$ and the chiral interaction energy, x. SS is a multiplier based on the senses of the two bonding CSAs. If the senses are not identical, the CSAs are *trans*; if the senses are the same, the CSAs are *cis*. The value of the SS multiplier for the various combinations of each pair of CSAs is found in Table 4.1.

The $Middle$ correction is made for rings which share a side with two other rings; it is based on the size and configuration of these rings. In $Middle$ rings, there will be at least two CSA pairs. These pairs of atoms can either interact with each other, in the same manner as the steric interaction, or with the just the ring, as an axial interaction. Stereoisomers of both tetradecahydroanthracene (Fig. 4.5a.) and of three rings with the middle ring having four ring-atoms ("middle-four"), such as [3]-ladderane (Fig. 4.5b) require explicit modeling due to the unique configurations of these stereoisomers. Table 4.2 lists the interactions of the various ring sizes and the $Middle$ energies they produce.

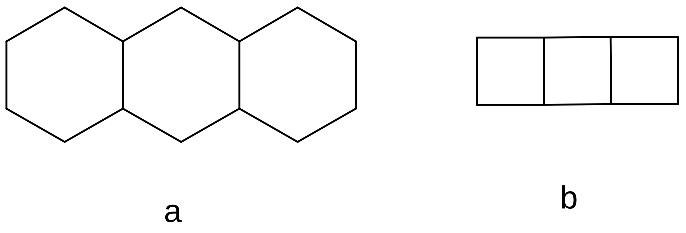

Figure 4.5: No stereochemistry specified in either a. Tetrahydroanthracene or b. Tricyclo[4.2.0.02,5]octane ([3]-ladderane)

Table 4.2: The *Middle* correction for various configurations of rings

Ring Size	**Ring Configuration / Senses**	**Middle Correction**
6-6-6	boat boat boat	$2y$
	chair boat chair	$4x$
	chair chair boat	x
	chair chair chair	0
X-4-X	xxxx	$4x$
X-X-X	xxxx	$2x$

Table 4.3: The RMS of the SPARC, PM3, and Group Additivity methods for the observed $\Delta\Delta H_f$ between pairs of stereoisomers.

Method	**RMS / pairs of stereoisomers**
SPARC	6.1 / 70
PM3	13.4 / 67
NIST/Benson Group Additivity	13.7 / 35

4.3 Data

The heat of formation of 169 compounds with chiral centers was used to develop this model. These measurements came from NIST[14] and the literature[15, 16]. To ensure the data is reliable, compounds measured before 1985 were obtained from references [15] and [16].

4.4 Results and Discussion

SPARC calculates the heat of formation by summing the value of each reaction center and its local perturbations and then analyzing the entire molecule for additional energy. It is more successful in modeling this property than either the semi-empirical PM3 method or Benson's group additivity method. In order to compare the effects of stereochemistry on the ΔH_f, the $\Delta\Delta H_f$ between stereoisomers is determined by taking the absolute value of the difference between the ΔH_f of the most stable stereoisomer and the ΔH_f of the other stereoisomer. Table 4.3 shows the overall RMS of these methods for the observed $\Delta\Delta H_f$ between pairs of stereoisomers.

The calculated versus observed values for compounds with linear chiral centers is shown in Figure 4.6. The average measured difference between sets of stereoisomers of linear chiral centers is 7.7 kJ mol^{-1}. As described in previously, SPARC does not differentiate between stereoisomers of linear chiral centers. Using no stereochemical modeling, the SPARC calculated $\Delta\Delta H_f$ RMS of molecules with linear chiral centers is 7.1 kJ mol^{-1}. PM3 differentiates between some stereoisomers with linear chiral centers and is unable to calculate a difference for others. Since PM3 works from an energy minimum, the reason for this differentiation is unclear. The RMS of the PM3 calculation for these molecules is 16 kJ mol^{-1}. The NIST version[12] of the Benson calculator produces surprising results for molecules with linear chiral centers. It is unable to calculate the ΔH_f for four sets of stereoisomers and only differentiates between two pairs of stereoisomers: 1) D-Arabinose and D-Ribose and 2) Beta-D-Fructose and L-Sorbose. Since Benson originally devised the *gauche* interaction to explain non-neighboring interactions this is unexpected. The other interesting result of the NIST calculator is the calculated difference in D-Arabinose and D-Ribose is 41 kJ mol^{-1} but the measured difference is only 7.6 kJ mol^{-1}. There are two *gauche* interactions in D-Arabinose and only one in D-Ribose, therefore the difference in these isomers, as predicted by Benson, should be only one *gauche* interaction or approximately 3.3 kJ mol^{-1}. There is also a single *gauche* interaction difference between Beta-D-Fructose and L-Sorbose but the calculated difference is also 41 kJ mol^{-1} while the actual measured difference is 5.2 kJ mol^{-1}. Why these molecules, when calculated by NIST, have such a large difference is unknown. The RMS of the NIST method for the calculation of seven sets of linear chiral center stereoisomers is 18.9 kJ mol^{-1}.

As shown in Figure 4.7, SPARC calculates a difference for thirty-five of the thirty-seven sets of single ring stereoisomers with interacting substituents. PM3 does so for only seven, and NIST does not calculate a difference for any sets of stereoisomers. The two sets of stereoisomers for which SPARC does not calculate a difference do not actually have interactions; their experimental difference is well within the reported measurement error of each stereoisomer. The RMS of the SPARC calculations is 5.6 kJ mol^{-1}, for PM3 the RMS is 10.8 kJ mol^{-1}, and for NIST the RMS is 11.7 kJ mol^{-1}. The average measured difference between stereoisomers in a single ring is 10.7 kJ mol^{-1}.

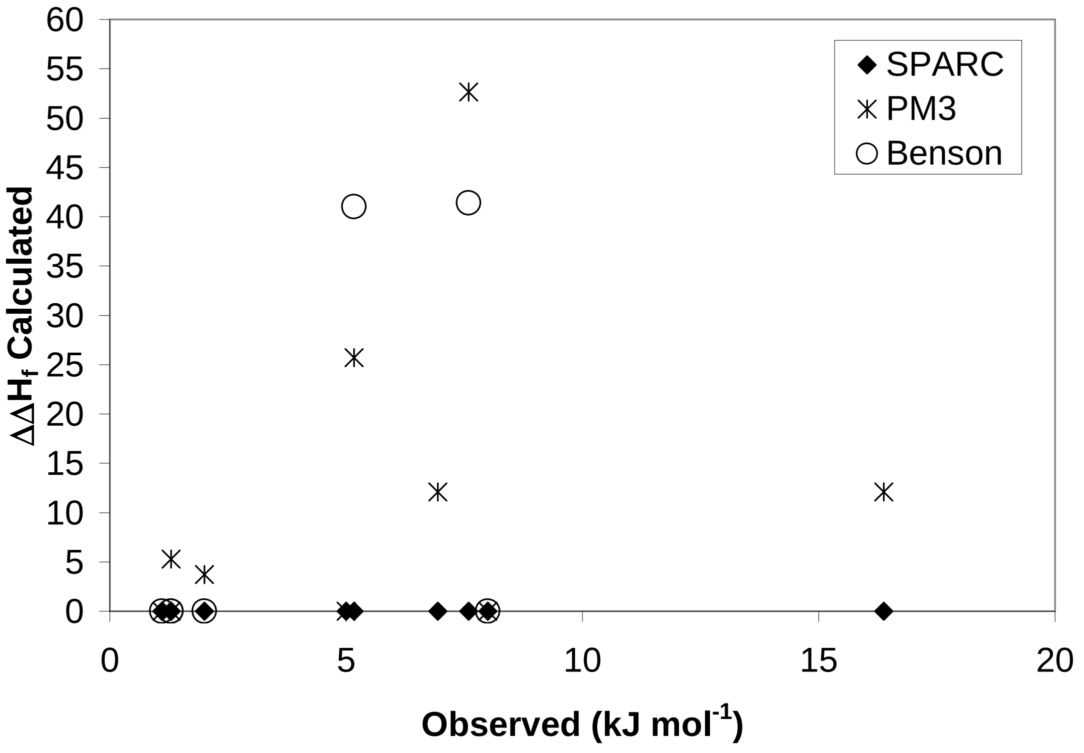

Figure 4.6: Linear Chiral Centers calculated versus observed values for 11 stereoisomers.

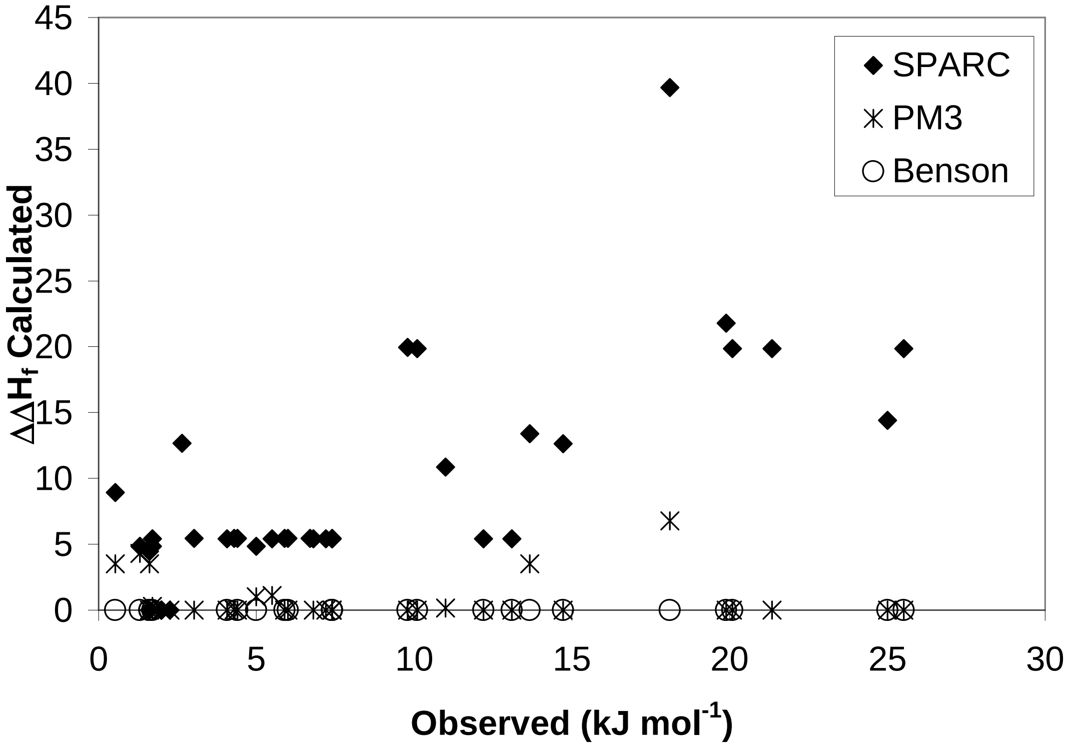

Figure 4.7: Calculated versus observed for 37 stereoisomers of chiral centers in single rings.

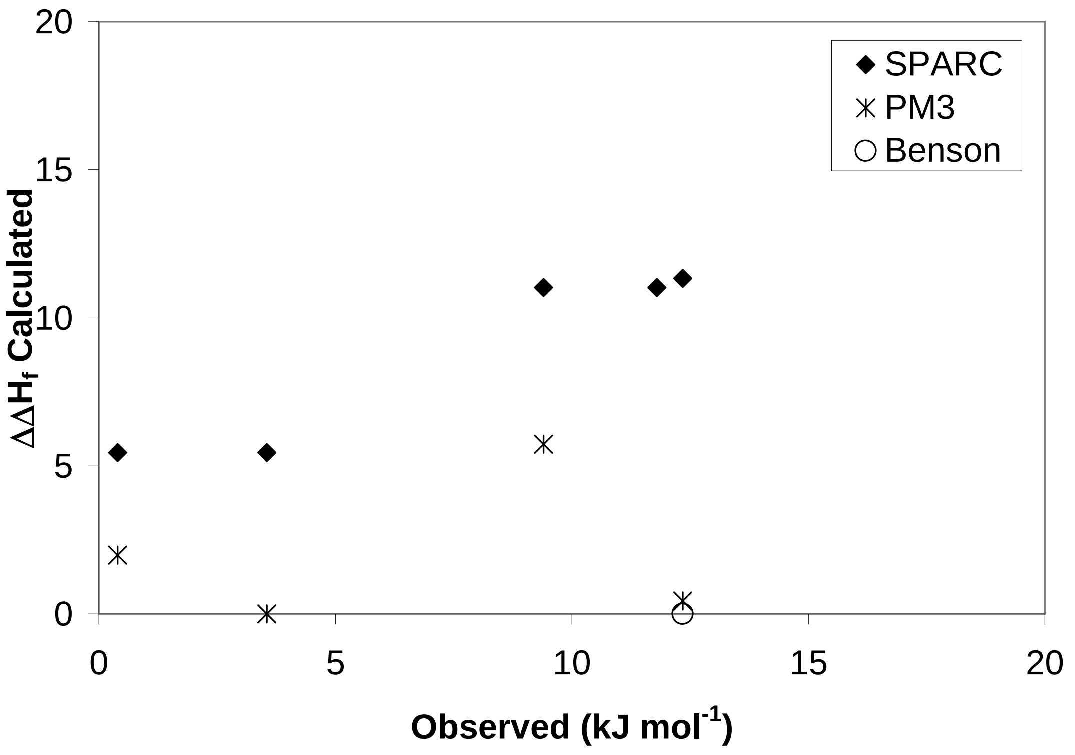

Figure 4.8: Bridge Rings Calculated versus observed for seven stereoisomers containing a bridge.

Figure 4.8 shows the calculated versus the observed values for the $\Delta\Delta H_f$ of the fourteen stereoisomers containing a bridge. SPARC calculates a difference in energy for all seven sets of stereoisomers; PM3 does so for only two sets and fails to calculate anything for one set of stereoisomers. NIST fails to compute twelve of the fourteen compounds and there is no difference in the one set of stereoisomers it was able to compute. The RMS of SPARC for these compounds is 3.9 kJ mol^{-1}. PM3 has a RMS of 18.4 kJ mol^{-1}, and the NIST RMS is 12.3 kJ mol^{-1}. The average measured difference between stereoisomer sets in molecules containing a bridge is 14.1 kJ mol^{-1}.

There are seventeen sets of stereoisomers containing sideshare chiral centers (Figure 4.9). SPARC calculates a $\Delta\Delta H_f$ between all seventeen sets of stereoisomers; PM3 does so for only two sets; NIST succeeds in calculating ten of the stereoisomers, but does not differentiate

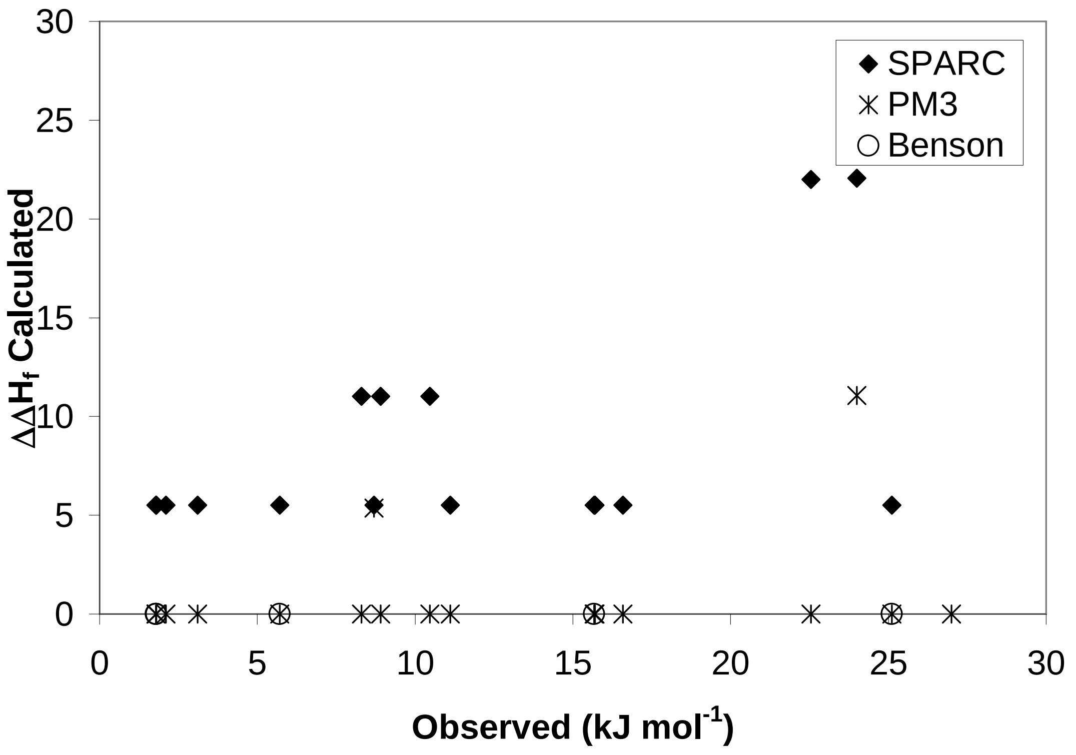

Figure 4.9: Calculated versus observed for the seventeen sideshare stereoisomers.

between any of them. For sideshare stereoisomers SPARC's RMS is 6.9 kJ mol^{-1}; the PM3 RMS is 13.8 kJ mol^{-1}; and NIST has an RMS of 13.5 kJ mol^{-1}. The average measured difference between stereoisomer sets in molecules containing sideshare chiral centers is 12.3 kJ mol^{-1}.

4.5 Conclusion

The stereocenter model was developed because of its importance in determining the overall heat of formation of an organic compound. The stereochemistry in a molecule can alter the ΔH_f by 5-25 kJ mol^{-1}. At room temperature, a shift in the enthalpy of reaction by 1 kcal mol-1 (4.184 kJ mol^{-1}) will generally change the equilibrium constant by a factor of five.[1] Therefore, for any calculation scheme to be accurate within a factor of ten, its uncertainty must be less than 9 kJ mol^{-1}. To meet the future goals of SPARC and accurately predict transition state energies, it is necessary to develop a heat of formation calculator to meet or exceed this requirement.

4.6 References

[1] Cohen, N.; Benson, S. W. J. Chem. Rev. 1993, 93, 2419-2438.

[2] Hilal, S. H.; Karickhoff, S. W.; Carreira, L. A.; Quant. Struct.-Act. Relat. 1993, 12, 389-396.

[3] Hilal, S. H.; Carreira, L. A.; Karickhoff, S. W. in Theoretical and Computational Chemistry, Quantitative Treatment of Solute/Solvent Interactions; Politzer, P.; Murray, J. S., Eds.; Elsevier Publishers: New York, 1994; pp 291-353.

[4] Hilal, S. H.; Karickhoff, S. W.; Carreira, L. A.; Quant. Struct.-Act. Relat. 1995, 14, 348-355.

[5] Hilal, S. H.; Karickhoff, S. W.; Carreira, L. A.; QSAR & Comb. Sci. 2003, 22, 565-573.

[6] Whiteside, T. S.; Carreira, L. A.; J. Theoretical and Computational Chemistry 2004, 3, 451-469.

[7] Whiteside, T. S.; Carreira, L. A.; submitted to Journal of Chemical Theory and Computation.

[8] Benson, S. W. Thermochemical Kinetics, 2nd ed.; John Wiley & Sons: New York, 1976.

[9] Stein, S. E.; Golden, D. M.; Benson, S. W. J. Phys. Chem. 1977, 81, 314-317.

[10] Stewart, J. J. P.; J. Comp. Chem. 1989, 10, 221-264.

[11] HyperChem(TM) Student Edition 7.0; Hypercube, Inc.; 1115 NW 4th St, Gainesville, Florida 32601.

[12] Stein, S. E.; Brown, R. L.; “Structures and Propeties Group Additivity Model” in NIST Chemistry WebBook, NIST Standard Reference Database Number 69, Linstrom, P. J.; Mallard, W. G., Eds; March 2003, National Institute of Standards and Technology: Gaithersburg MD, 20899 (http://webbook.nist.gov).

[13] Eremenko, L.T; Korolev, A.M; Berezina, L.I.; Kirpichev, E.P.; Rubtsov, Yu.I.; Sorokina, T.V. Bull. Acad. Sci. USSR, Div. Chem. Sci. 1985, 795-798.

[14] Afeefy, H. Y.; Liebman, J. F.; Stein, S. E.; “Neutral Thermochemical Data” in NIST Chemistry WebBook, NIST Standard Reference Database Number 69, Linstrom, P. J.; Mallard, W. G., Eds; March 2003, National Institute of Standards and Technology: Gaithersburg MD, 20899 (http://webbook.nist.gov).

[15] Cox, J. D. and Pilcher, G. Thermochemistry of Organic and Organometallic Compounds; Academic Press: London, 1970.

[16] Pedley, J. B.; Naylor, R. D.; Kirby, S. P. Thermochemical Data of Organic Compounds, 2nd ed.; Chapman and Hall: London, 1986.

Chapter 5

Other Modeling

5.1 Introduction

After the completion of the hydrocarbon model, it was thought the next simplest compounds to model would be those containing oxygen. This was assumed because in the combustion reaction

$$C_xH_yO_z \longrightarrow H_2O + CO_2 \tag{5.1}$$

there are no by-products, such as N_2, $H_2SO_4(nH_2O)$, or $HCl(nH_2O)$ which have to be accounted when combusting compounds containing other elements, as described by Cox and Pilcher.[9] With this in mind, we attempted to model alcohols, aldehydes, and ketones.

5.2 Alcohols

Pure alcohols are those compounds which contain Carbon, Hydrogen, and one or more OH-groups. We learned when modeling hydrocarbons that molecules without rings or aromatic structures are the simplest to model. Our attempt at modeling alcohols followed this reasoning and alcohols in chains were easily modeled. The contribution of an alcohol group remains constant and is independent of chain length or branching. These compounds have an RMS of 3.78 kJ mol^{-1}.

It was assumed that diols (molecules with two alcohol groups) would be similiarly modeled. However, as the results were examined, it appeard that the assumption that each alcohol group contributed a constant amount appeared invalid. A plot of diol length versus heat of formation was not linear. The longer the hydrocarbon chain of the backbone, the more scattered the data became. We thought at first that this could be explained through hydrogen bonding; however, intra-molecular hydrogen bonding would only occur in small molecules. The scattering in the data is probably due to diols being hygroscopic and samples, if not kept under nitrogen, are likely to be contaminated with water or perhaps dissociated into other compounds. Since we could not model alcohol chains with the degree of accuracy necessary for SPARC, it was decided to focus our efforts on more stable groups.

5.3 Aldehydes

There are suprisingly few heat of formation measurements for aldehydes, only eight. As an aldehyde functional group is considered a reaction center in SPARC, all that was necessary was to add a parameter to SPARCs database which determines how much an aldehyde contributes to the heat of formation. This parameter has a value of -168.37 kJ mol^{-1}. The RMS for these compounds is 2.6 kJ mol^{-1}.

5.4 Ketones

Ketones are more difficult to model than aldehydes, although they have the same reaction center. This is because the ‘oethylenic’-group, has three possible steric interactions (similar to an ethylenic), as opposed to one in an aldehyde, and has dipole effects. The same parameter was used as for aldehydes, and for chains the model worked well, producing an RMS of 4.21 kJ mol^{-1} for 27 compounds.

There was no data for oethylenics and ethylenics nor was there data for oethylenics and acetylenics bonded together. When we attempted to model ketones in rings, the modeling done for oethylenics in chains no longer worked. We sufferd from the problems of large amounts of bad data and small data sets for each supposed effect. For these reasons we moved on to calculating the heat of formation of halogens.

5.5 Assorted projects

Other projects were also undertaken to improve the SPARC modeling system as a whole. This included updating 'distance.pro' and 'pkadata.pro' to handle any arbitrary aromatic compound and to calculate 'LeninLenout' data. LeninLenout is the distance from a substituent on an aromatic reaction center to another substitent on the same reaction center. This information is necessary when determining resonance contribution of substituents through a ring. As aromatic reaction centers can be expressed as hexagons, we used cartesian coordinates to generate a map of the compounds and calculate the distance and angles between each atom. We also separated the benzene, napthalene, anthracene, and phenanthrene templates so each *could* be trained independently of the others.

During this time we also moved SPARC from ALS to SICSTUS Prolog. This was done because ALS Prolog is no longer being maintained and SICSTUS is one of the fastest and highest quality Prolog distributions available.

Chapter 6

Tools

6.1 Introduction

The process of model building is expedited when the proper tools are provided. Lucinda Bornander created many tools to: add new data to the SPARC database, remove training sets from the database, build training sets from data in the database, and search the database for property values. A large number of these were originally written in Visual Basic, but were converted to Cold Fusion, providing a platform independent method of processing data. This "toolbox" has been expanded upon and new tools added to ease the implementation of SPARC and facilitate model development.

These new tools include: database management, improved training file building, viewing of SPARC data, user area creation, and quality control of the SPARC models. The code for these tools is found in Appendix A.

6.2 The SPARC Database

Originally, SPARC's data was kept in multiple Microsoft Access databases. In order to ease the maintenance of the data, we ported all of it into one MySQL database called *sparcdb*. MySQL has the dual advantages, over Microsoft Access, of faster record searching and allowing larger queries to be built.

In order to create one database, the various tables in the other databases were imported into the *sparcdb*. It was also desired to have a consistent table layout, as far as possible. With these considerations in mind, the *pka*, *physical properties*, *heat of formation*, and *entropy of* *fusion* tables were imported and updated as necessary. The training file database was also inserted. The *indexMaster* table was created to build training files, link data and provide a common source for unique SMILES strings. This is the largest table in the SPARC database and contains all of the compounds used in SPARC. It lists the CAS number, unique SMILES string, name, substituents, reactophores, molecular weight, and generics of each molecule.

While the SPARC database is relatively static, it still requires maintenance. Whenever new data or compounds are discovered, these must be added to the appropriate table. For heat of formation and entropy of fusion data this is accomplished via a Cold Fusion tool which allows the user to select which table to insert data and enter the information for the appropriate fields. The code for inserting heat of formation data is included in Appendix A. Code for the entropy of fusion is not included because it is nearly identical. No tools were created to insert data for the other properties because these models are relatively mature; however, if new data is found, it can be inserted into the database by hand. If a new compound is obtained, it is necessary to update *indexMaster* as well. Since *indexMaster* contains over 81,000 compounds and has many fields, the simplest way to add compounds is to create a .mol file named "CAS_number.mol" and place it in the folder `c:\Inetpub\wwwroot\TadsTool\Hf_Database_Builder\Mol\Temp` on Ibmlc2. Once the file is in this location a Cold Fusion tool (Appendix A, fixmol.cfm) is used to create the unique smiles string, calculate the molecular weight, find the substituents, reactophores, and generics of the compound and update indexMaster.

Since the SPARC database contains "real-world" data, some erroneous data may inadvertently be included in the database. Because some of this data is to be kept for further analysis, a method for "commenting it out" was developed (Appendix A, remove.cfm and remove_2.cfm). Also, because the heat of formation, entropy of fusion, and melting point data were obtained from the NIST chemical webbook, which has known errors, this tool was written specifically to "comment out" data from these tables. When a user wishes to "comment out" data, a field in the appropriate table is updated and the next time a training file is built, this compound will be commented out.

Some compounds in indexMaster do not have CAS numbers or names associated with them; an ongoing project is to update this table with the CAS numbers and names. This is done by taking a SMILES string without a CAS number and generating a .mol file from it, then pasting the .mol file into SciFinder. If the compound is found, indexMaster can be updated with the name and CAS number. To expedite this process, a tool was built as an administrator function in the Bornander training file tool. The code for this process is found in Appendix A, fix_indexmaster.cfm and get_cas2.cfm.

6.3 Training File Tools

The training file tools were initially developed by Bornander for producing files to train pka and GC data; with the addition of heat of formation and entropy of fusion it was desired to make this into a generic process. This generic training file builder was constructed by modifying the code to query the appropriate table in the unified SPARC database. Bornander's code was functionally complete for all data sets. Missing was a method to sort the file by type and subtype of molecule: alkane chains, alkene chains, compounds with one ring, compounds with two rings, aromatic compounds, etc. To correct this, a new file builder was created; included in this new method was a way to "comment out" compounds and include multiple measurements of data for the same compound. If a compound is marked as "commented out" in the database, the training file builder will include that data in the training file, but will comment it out. If there are multiple measurements of the same data, these are ranked, in the database, according to preference. The training file builder will insert all of this data into the training file with the "best" value as the value to be trained and the remaining values commented out. By editing the database, the rank can be changed and the next time the training file is built this new rank will be first. The code for this improved training file builder is included in Appendix A.

Another useful tool in building training files is one which converts values from the SPARC data area into training file parameters. When a model is in the development process there are many new parameters being added. This tool takes the data and parses it to build training file parameters. This allows parameters to be quickly and accurately built for inclusion in training files. This code is found in Appendix A as data2param_top.cfm and data2params.cfm.

6.4 SPARC Data Viewer / Comparer

After SPARC models are built and trained, it is helpful to examine the results of the model and compare them to other methods and the observed values. The data viewer shows the molecular structure as well as the observed and calculated values of the molecules. The viewer developed specifically for the heat of formation, besides displaying the difference in the calculated and observed values and CAS number, also shows the reference of the observed data and the command necessary to run the model the compound. At the bottom of the file, the RMS between the calculated and observed values and the actual RMS, as obtained from the error in the actual measurements, is displayed. In the heat of formation data comparer tool, the observed data is compared with the SPARC, PM3, and Benson/NIST calculated values. Also shown are the RMS' for these various methods and the total number of compounds calculated by each. This tool is extremely useful to check the linearity of the calculated values. The Cold Fusion code for these two tools is listed in Appendix A. Only the heat of formation version of this code is listed, as the generic versions of this code are nearly identical.

6.5 Quality Control of SPARC

Quality control is an important process of model development. When one portion of a model changes, it is necessary to know how this change affects other sections of the model. A quality control program was developed to alert SPARC developers when a model change occurs.

The quality control program examines the model every night and sends an email to the SPARC developers when its task is completed. To check the model, the quality control script runs the aqueous pka, heat of formation, physical properties, hydrolysis, electron affinity, diffusion, ehalf, nonaqueous pka, and hydration batch files. These files test all of the "production level" SPARC models. Each night, after the batch files have been tested, the difference in the current output file and the previous day's output file is calculated. If there are differences in the two files, SPARC sorts the differences into large differences and small differences. Large differences are those molecules which have changed by more than 10% from the previous day; small differences are those that have changed by less than this amount. Once the differences have been calculated an email is sent to the SPARC developers, letting them know of any changes and which molecules were affected. If the model has not changed, this is stated in the email. A daily notification ensures the quality control program actually worked and a flaw in a model did not cause the whole process to fail.

6.6 Common User Area

Before a new developer can work with the SPARC system, their work area must be created. For SPARC to run properly each user needs multiple directories and files in their home directory. Previously, to add a new user to the system, an existing area was copied and renamed. This resulted in files with permission problems and unwanted files being duplicated. To improve on this process we developed a shell script that generates a user's work area and then removes itself. This script, "build_area", is found in the SPARC area inside of the "user_template" directory. To build the user's area, this script is copied to the user's home directory and then run. It copies or creates all of the necessary files and creates the necessary directories; finally, the script removes itself, leaving the user with a freshly created area.

6.7 Summary

A full code listing, broken down by section, can be found in Appendix A. These tools facilitate adding users to the SPARC development process, managing SPARC model development, building training files, and analyzing the data from SPARC model development.

Chapter 7

Conclusions

7.1 Review of Findings

The SPARC ΔH_f model is nearly as accurate as most experimental measurements of hydrocarbons and halogenated hydrocarbons. It has been tested with all reliably measured compounds. The structures of these compounds range from hydrocarbon chains to highly branched perfluoroalkanes; from conjugated rings to halogenated aromatics. SPARC provides a quick and reliable method for determining the ΔH_f of these compounds and a more accurate alternative to using semi-empirical or group additivity methods.

Information on the effects which contribute to the ΔH_f was acquired and a database of high quality, measured compounds was compiled before the a SPARC ΔH_f calculator was developed. After this background knowledge was obtained the models were built.

The SPARC model of the heat of formation uses procedures developed for other SPARC models and has similarities between semi-empirical and group additivity methods. The heat of formation is expressed as a function of the energy required to add an atom to the free element state. It represents the energy difference between the free element state and the whole molecule state. This energy difference is modeled by expressing ΔH_f in terms of the summation of the contributions of all the components, perturber(s), and reaction centers(s), in the molecule (Equation 7.1).

$$\Delta H_f = \sum_{C=1}^{n} [(\Delta H_f)_C + A_C] + (\Delta H_f)_{Whole\_molecule} \tag{7.1}$$

Using the standard SPARC procedure, molecules are first broken into C reaction centers (essential atoms/groups) and their perturber structures. In this equation, $(\Delta H_f)_C$ is the intrinsic heat of formation of the reaction center, C, and is assumed to be unperturbed and independent of any perturbing structures. A_C describes adjustments made to the $(\Delta H_f)_C$ by the perturber structure(s); the $(\Delta H_f)_{Whole\_molecule}$ is any additional energy added (or subtracted) due to the overall molecular structure, e.g. distribution of NBMO charge in conjugated systems.

The reaction center perturbation, A_C, is further factored into mechanistic components SPARC calculates for perturbations to the heat of formation of each reaction center, as described by Equation 7.2.

$$A_C = A_{steric}(\Delta H_f)_C + A_{resonance}(\Delta H_f)_C + A_{connectivity}(\Delta H_f)_C \tag{7.2}$$

$A_{steric}(\Delta H_f)_C$, $A_{resonance}(\Delta H_f)_C$, and $A_{connectivity}(\Delta H_f)_C$ describe how the size, resonance, and connectivity of the perturber(s) affect the reaction center's value. The steric interaction is the effect of a perturber's size on the reaction center. The resonance interaction is a description of the amount of stabilization a molecule undergoes due to charge being distributed out of the reaction center and into the perturbing structures. The connectivity adjustment describes the change in a reactophore's energy when a hydrogen atom of a base reactophore is replaced with another group.

The "whole molecule" correction, $(\Delta H_f)_{Whole\_molecule}$, is the sum of corrections applied to the entire molecule; this includes corrections for rings, additional resonance, and additional steric energy (Equation 7.3).

$$(\Delta H_f)_{Whole\_molecule} = (\Delta H_f)_{rings} + (\Delta H_f)_{xres} + (\Delta H_f)_{xsteric} \tag{7.3}$$

The number, size, and connectivity of the rings in a molecule determine the amount of correction for rings, $(\Delta H_f)_{rings}$. There is an additional resonance correction, $(\Delta H_f)_{xres}$, for homoaromaticity. $(\Delta H_f)_{xsteric}$ models the strain induced by various spatial environments. All of the perturbations and adjustments are empirically trained against experimentally determined values of the heat of formation.

The hydrocarbon model underwent extensive development because hydrocarbons have the most reliably measured heats of formation and also because these compounds provide the basic structural framework for all other types of organic compounds. The 587 hydrocarbons used to build this model have a SPARC calculated RMS of 4.50 kJ mol^{-1}.

The halogenated hydrocarbon model was developed because these halogens can only bond to one other reaction center. Also, other researchers [5, 27, 22, 26] had successfully modeled this group of compounds using group additivity. Unfortunately, this model requires a large number of parameters to correctly model all of the measured compounds. Halogenated hydrocarbons have a calculated RMS deviation of 5.18 kJ mol^{-1} for 202 compounds.

The stereocenter model was developed because stereochemistry in a molecule can alter the ΔH_f by 5-25 kJ mol^{-1}. 169 stereocompounds (70 pairs of stereoisomers) were used to build the model, SPARC calculates a 6.1 kJ mol^{-1} RMS deviation between pairs of stereoisomers ($\Delta\Delta H_f$).

At room temperature, a shift in the enthalpy of reaction by 1 kcal mol-1 (4.184 kJ mol^{-1}) will generally change the equilibrium constant by a factor of five.[5] Therefore, for any calculation scheme to be accurate within a factor of ten, its uncertainty must be less than 9 kJ mol^{-1}. To meet the future goals of SPARC and accurately predict transition state energies, it is necessary to develop a heat of formation calculator to meet or exceed this requirement.

7.2 Future Studies

While at present SPARC only calculates the heat of formation of hydrocarbons and halogenated hydrocarbons, work is in progress to extend the capability of these models to calculate the ΔH_f for molecules containing oxygen, nitrogen, phosphorus, and sulfur. Once the heteroatom models are complete it will be possible to integrate them into SPARC's kinetic and redox rate models in order to better predict transition state energies. The extrapolatability of models to other types of chemistry is one of SPARC's greatest strengths and is the impetus behind continued development of these models.

A complete listing of the compounds used to develop these models can be found in Appendix B. The code developed for modeling the heat of formation can be found in Appendix C.

Bibliography

[1] Hilal, S. H.; Carreira, L. A.; and Karickhoff, S. W. "Estimation of Chemical Reactivity Parameters and Physical Properties of Organic Molecules Using SPARC." In *Quantitative Treatment of Solute/Solvent Interactions*; Politzer, P. and Murray, J. S., Eds.; Theoretical and computational chemistry; Elsevier: New York, 1994; Vol. 1, 291–353.

[2] Lowry, T.H. and Richardson, K.S. *Mechanism and Theory in Organic Chemistry. 3rd edition.* Harper and Row: New York, 1987.

[3] Taft, R.W. *Progress in Organic Chemistry*, John Wiley & Sons: New York, 1987; Vol. 16.

[4] Atkins, P. *Physical Chemistry.*, 5th edition; WH Freeman and Company: New York, 1994; 85.

[5] Cohen, N.; Benson, S. W. *J. Chem. Rev.* **1993**, *93*, 2419–2438.

[6] *Computational Thermochemistry.*Irikura, K. and Frurip D., Eds.;American Chemical Society: Washington DC, 1998.

[7] Benson, S.W. *Thermochemical Kinetics*, 2nd edition; John Wiley & Sons: New York, 1976.

[8] Afeefy, H.Y.; Liebman, J.F.; and Stein S.E. "Neutral Thermochemical Data" in NIST Chemistry WebBook, NIST Standard Reference Database Number 69, Eds. P.J. Linstrom and W.G. Mallard, July 2001, National Institute of Standards and Technology, Gaithersburg MD, 20899 (http://webbook.nist.gov).

[9] Cox, J. D. and Pilcher, G.; *Thermochemistry of Organic and Organometallic Compounds*; Academic Press: London, 1970.

[10] Pedley, J. B.; Naylor, R. D.; and Kirby, S. P.; *Thermochemical Data of Organic Compounds*, 2nd ed.; Chapman and Hall: London, 1986.

[11] Hilal, S. H.; Karickhoff, S. W.; Carreira, L. A.; *Quant. Struct.-Act. Relat.* **1995**, *14*, 348–355.

[12] Hilal, S. H.; Karickhoff, S. W.; Carreira, L. A.; *QSAR & Comb. Sci.* **2003**, *22*, 565–573.

[13] Hilal, S. H.; Karickhoff, S. W.; Carreira, L. A.; *Quant. Struct.-Act. Relat.* 1993, *12*, 389–396.

[14] Whiteside, T. S.; Carreira, L. A.; *J. Theoretical and Computational Chemistry* 2004, *3*, 451–469.

[15] Whiteside, T. S.; Carreira, L. A.; submitted to *Journal of Chemical Theory and Computation.*

[16] Carreira, L. A.; SPARC. http://ibmlc2.chem.uga.edu/sparc

[17] Stein, S. E.; Golden, D. M.; Benson, S. W. *J. Phys. Chem.* **1977,** *81*, 314–317.

[18] Liebman, J. F.; Paquette, L. A.; Peterson, J. R.; Rogers, D. W.; *J. Am. Chem. Soc.* **1986**, *108*, 8267–8268.

[19] Stewart, J. J. P.; *J. Comp. Chem.* **1989**, *10*, 221–264.

[20] HyperChem(TM) Student Edition 7.0; Hypercube, Inc.; 1115 NW 4th St, Gainesville, Florida 32601.

[21] Roth, W. R.; Adamczak, O.; Breuckmann, R.; Lennartz, H. W.; Boese, R.; *Chem. Ber.* **1991**, *124*, 2499–2521.

[22] Kolesov, V. P.; Papina, T. S.; *Russian Chemical Reviews* **1983**, *52*, 425–439.

[23] Paddison, S. J.; Tschuikow-Roux, E.; *Int. J. Thermophysics* **1998**, *19*, 719–730.

[24] Smart, B. E. in *Molecular Structure and Energetics* Vol 3.; Liebman, J. F.; Greenberg, A., Eds.; VCH Publishers: Deerfield Beach, FL, 1986; 141–191.

[25] Stein, S. E.; Brown, R. L.; "Structures and Propeties Group Additivity Model" in NIST Chemistry WebBook, NIST Standard Reference Database Number 69, Linstrom, P. J.; Mallard, W. G., Eds; March 2003, National Institute of Standards and Technology: Gaithersburg MD, 20899 (http://webbook.nist.gov).

[26] Kolesov, V. P.; Kozina, M. P. *Russian Chemical Reviews* **1986**, *55*, 912–928.

[27] Joshi, R. M. *J Macromol Sci-Chem* **1974**, *A8*, 861–885.

[28] Eremenko, L.T; Korolev, A.M; Berezina, L.I.; Kirpichev, E.P.; Rubtsov, Yu.I.; Sorokina, T.V. *Bull. Acad. Sci. USSR, Div. Chem. Sci.* **1985**, 795–798.

Appendix A

SPARC Tools

A.1 Database management tools

A.1.1 Insert_hf.cfm

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">

<html>
<head>
<title>Add data to HF</title>
</head>

<body>
<cfoutput>
<form action="insert2_hf.cfm?db=#db#" method="post">
<table>
<tr><td>Cas number</td><td><input type="text" name="cas"></input></td></tr>
<tr><td>DfH</td><td> <input type="text" name="dfh"></input></td></tr>
<tr><td>Error </td><td> <input type="text" name="error"></input></td></tr>
<tr><td>Method</td><td> <input type="text" name="m"></input></td></tr>
<tr><td>Refs</td><td> <input type="text" name="refs"></input></td></tr>
<tr><td>Comments </td><td><input type="text" name="c"></input></td></tr>
<tr><td>Rank </td><td> <input type="text" name="rank"
value = 1></input></td></tr>
</table>
<input type="hidden" name="table" value = "dataHF"></input>
<p>
<input type="submit" name="button" value="Submit">
</form>

</cfoutput>
<hr>
<center><a href="top_page_DB.cfm">Home</a></center>
</body>
</html>
```

A.1.2 Insert2_hf.cfm

```
<cfset newdfh = dfh * 1>
<cfset newerror = error * 1>

<!--- update the ranks --->
<cfquery datasource=#db# name="ru">
SELECT id,Rank FROM #table#
WHERE cas = #cas#
</cfquery>

<cfloop query="ru">
<cfset newrank = #Rank#+1>
<cfquery datasource=#db# name="nru">
UPDATE #table#
SET Rank = #newrank#
WHERE id = #id#
</cfquery>
</cfloop>

<!--- add the new data --->
<cfquery datasource=#db# name="insert">
INSERT INTO #table#(cas,DfH,Error,Method,Refs,Comments,Rank)
VALUES(#cas#,#newdfh#,#newerror#,'#m#','#refs#','#c#',#rank#)
</cfquery>

<cflocation url="insert_hf.cfm?db=#db#">

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">

<html>
<head>
<title>Untitled</title>
</head>

<body>

<cfoutput>

</cfoutput>
</body>
</html>
```

A.1.3 fixmol.cfm

```
<!---- Table to update ---->
<cfset table = "indexMaster">
<!---- Change to point to the mol area to be read ---->
<cfset molpath='c:\inetpub\wwwroot\tadstool\hf_database_builder\mol\temp\'>
<cfset movepath='c:\inetpub\wwwroot\tadstool\hf_database_builder\mol\'>
<cfdirectory  directory="#molpath#" name="mydirectory"  sort="name DESC">
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head><title>Gets Unique Smiles From Mol</title></head>
<body>
<!--- create the smiles class for calculation --->
<cfobject action="CREATE" name="MySmilesObject"
class="NewCD_Proj.SmilesCF" type="COM">
<!--- create a sun class for calculation --->
<cfobject action="CREATE" name="MyObject"
class="NewCD_Proj.clsSun" type="COM">
<cfset MyObject.SunHost = "ibmlc2.chem.uga.edu">
<cfset MyObject.SunPort = 1250>
<cfset MyObject.ServerType = "new">
<cfset connected=MyObject.Connect()>
<cfif not connected>
<cflocation url="error.cfm?error=1">
</cfif>
<cfset j=0>
<table border=1>
<tr><td><b>File</b></td><td><b>CAS</b></td><td><b>Smiles</b></td></tr>
<cfoutput query="mydirectory">
<cfif trim(name) is ''>
<cfelseif trim(name) is '.'>
<cfelseif trim(name) is '..'>
<cfelse>
<cfset cas=listfirst(name,".")>
<cfset molfilename=molpath & #name#>
<cfset calculateOK=
MySmilesObject.SmilesFromMol(molfilename,0,"UniqueSmiles","MyErrorMessage")>
<tr><td>#name#</td><td>#cas#</td>
<cfif calculateOK>
<td>#uniquesmiles#</td>
<cfset mysmiles=uniquesmiles>
<cfset SmilesIn='#mysmiles#'>
<cfset MW="None">
<cfset CAS2="None">
```

```
<cfset MyName="None">
<cfset Subs="None">
<cfset Reacs="None">
<cfset the_generic="None">
<cfset the_generic=MyObject.get_generic(SmilesIn)>
<cfset calculateOK2=
MyObject.Prop_Cas(SmilesIn,"MW","CAS2","MyName","Subs","Reacs")>
<cfset CAS2 = 0>
<cfset MyName = "???">
<cfif calculateOK2 and MW neq ''>
<cfquery datasource = "#db#" name = "myupdate">
UPDATE #table#
SET MolWt=#MW#,
Substituents='#Subs#',
Reactophores='#Reacs#'
WHERE cas = #cas#
</cfquery>
   <cfif the_generic is not 'fail'>
<cfquery datasource = "#db#" name = "myupdate2">
  update #table#
  set generic='#the_generic#'
     where cas = #cas#
</cfquery>
</cfif>
<cfelse>
<cfquery datasource = "#db#" name = "myupdate">
Update #table#
Set substituents='bad'
Where cas = #cas#
</cfquery>
</cfif>
<cfelse>
<td><b>Failed</b></td>
<cfset mysmiles="Failed">
</cfif>
<cfquery datasource="#db#" name="up">
UPDATE #table#
SET smiles = '#replace(mysmiles,'\','\\',"ALL")#'
WHERE cas= #cas#
</cfquery>
  </tr>
<cffile action="MOVE" source="#molfilename#" destination="#movepath#">
</cfif>
</cfoutput>
<cfset rc=MyObject.disconnect()>
```

```
</table>
</body>
</html>
```

A.1.4 REMOVE.CFM

```
<head>
<title>Mark out data</title>
</head>

<body>
<cfoutput>
<form action="remove_2.cfm?db=#db#" method="post">
Select the table in you wish to mark out data.
<cfset names="dataHF,dataSfus,dataSPARC,dataTfus">
    <select name="table">
<cfloop index="x" list="#names#">
<option>#x#</option>
</cfloop>
    </select>
<br>
<br>
Select the cas number to mark out.<input type="text"
name="cas"></input><br>
<br>
Enter the comment for the comments2 column.<input type="text"
name="c2"></input>
<p>
<input type="submit" name="button" value="Submit">
</form>

</cfoutput>
<hr>
<center><a href="top_page_DB.cfm">Home</a></center>
</body>
</html>
```

A.1.5 REMOVE_2.CFM

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">

<html>
```

```
<head>
<title>Untitled</title>
</head>

<body>
<cfoutput>
<cfif cas is not "" AND c2 is not "">

<cfquery datasource=#db# name="markout">
UPDATE #table#
SET xuse = 'n'
WHERE cas=#cas#
</cfquery>

<cfquery datasource=#db# name="markout">
UPDATE #table#
SET comments2 = '#c2#'
WHERE cas=#cas# AND rank = 1
</cfquery>
OK!
<a href="remove.cfm?db=#db#">Remove</a>
<cfelse>
Go back and fill in completely!
<a href="remove.cfm?db=#db#">Remove</a>
</cfif>
</cfoutput>
</body>
</html>
```

A.1.6 FIX_INDEXMASTER.CFM

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">

<html>
<head>
<title>Fix indexMaster</title>
</head>
<body>

<cfoutput>

<cfif id is 1>
The smiles is #smiles#<br>
```

```
<br>

<cfset myfile= "\\ibmlc2\wwwroot\Born\cas_todo\" & "#smilesid#" & ".mol">
#myfile#<br>
<br>
<form action=fix_indexMaster.cfm>
<input type = text name="cas"> CAS Number from SciFinder<br><br>
<input type = text name="name"> Name from SciFinder<br><br>
<input type = "checkbox" name="notin" value="true">
Check this if cannot find in SciFinder<br><br>
<input type = hidden name="id" value = "2">
<input type = hidden name="smiles" value = '#smiles#'>
<input type = hidden name="myfile" value = '#myfile#'>
<input type = submit value = "Submit" >
</form>

<cfelseif id is 2>
  <cfif NOT IsDefined("notin")><cfset notin="false"></cfif>
  <cfif #notin# is "true">
   <cfquery datasource=#application.datasource# name="fix">
       UPDATE indexMaster
       SET comments = "not found in SciFinder"
       WHERE  smiles='#replace(smiles,'\','\\',"ALL")#'
       </cfquery>
<cflocation url="get_cas2.cfm">
  <cfelse>
  <cfset cas = replace(cas,'-','',"ALL")>
  <cfquery datasource=#application.datasource# name="check">
   SELECT * FROM indexMaster
WHERE cas = #cas#
  </cfquery>
         <cfif check.recordcount is 0>
       <cfquery datasource=#application.datasource# name="fix">
       UPDATE indexMaster
       SET cas = #cas#, name = '#name#'
       WHERE  smiles='#replace(smiles,'\','\\',"ALL")#'
       </cfquery>
<cffile action="delete" file = "#myfile#">
Database Updated!
<cflocation url="get_cas2.cfm">
       <cfelse>
         Sorry buddy this cas number is already in the database for:<br>
         #check.smiles#<br>
         This is your smiles:<br>
         #smiles#<br>
         Why do they have the same cas?  Something is wrong.
       </cfif>
   </cfif>
</cfif>
</cfoutput>
<hr><br>
<Font size = +2><center><a href=get_cas2.cfm>
Get_CAS2</a></center></font><br>
<Font size = +2><center><a href=admin_menu.htm>
Admin</a></center></font>
</body>
</html>
```

A.1.7 get_cas2.cfm

```
<cfset table = "indexMaster">
<!--- Get all the records that are not in this database --->
<cfquery datasource = #application.datasource# name= "notin">
SELECT * FROM #table#
WHERE cas = -2
ORDER BY id
</cfquery>
<html>
<head> <title>SMILES not in db</title> </head>
<body>
<font size=+1>
If you would like to avoid doing any updating, then click
<a href="get_cas.cfm?x=1"> here</a> to go back to "get_cas.cfm"
and have it add the CAS,MW,etc.</font><br><hr>
<cfoutput>
Clicking on a smiles string will take you to a page where you
can copy the path, go to SciFinder and paste that path into the
structure import section, be sure to change the file type to *.mol,
and then click "get structure".  When you do this the proper
compound should appear and you can get the CAS number and name.<br>
MAKE SURE TO REPLACE BENZENE'S WITH PROPER BENZENE'S, THE SMILES
TO MOL PROGRAM MESSED THEM UP.<br><br>
If multiple compounds appear, then pick the one that is a "real compound",
not a component of a multi component system or a "radical".
This is usually found near the bottom of the list. <br><br>
If it appears that something is already in the database or
you are not sure which cas to use, ask Tad or Butch,
probably Butch, what to do.<br>
```

```
<br>
If you have decided that the structure is wrong,
you will need to edit indexMaster by hand --
deleting the old smiles and entering the new one. (
I suppose this could actually be a page also..but I'm lazy. -Tad)
<hr>
This is the list of #notin.recordcount#
SMILES that do not have CAS numbers associated with them:<br>
<br>
<cfloop query = "notin">
<cfset newsmiles = urlencodedformat(#smiles#)>
#id#<a href=
"fix_indexMaster.cfm?id=1&smiles=#newsmiles#&smilesid=#id#">#smiles#
</a> #comments#<br>
</cfloop>
</cfoutput>
<hr>
<Font size = +2><center><a href=admin_menu.htm>Admin</a></center></font>
</body>
</html>
```

A.1.8 TS_Viewer.cfm

```
<cfinclude template="../login_check.cfm">

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">

<html>
<head>
<title>Sparc Data Viewer</title>
</head>

<body>
<font color="#7f1111"  size="5" face="trebuchet ms,arial,helvetica">
Sparc Data Viewer</font>
<p>
<form action="Choice.cfm" method="post">
<input type="radio" name="TS" value="Display Joe Diff" checked>
Display Joe Diff<br>
<input type="radio" name="TS" value="Display Generic Joe Diff">
Display Generic Joe Diff<br>
<input type="radio" name="TS" value="Display Training Set">
Display Training Set<br>
```

```
<input type="radio" name="TS" value="Upload"> Upload File<br>
<input type="radio" name="TS" value="Delete"> Delete File <p>
<input type="submit" value="Submit">
</form>
Take me back to the <a href="../tadtool.cfm">Research Tools Portal</a>.
</body>
</html>
```

A.1.9 ChooseSetDiff.cfm

```
<cfinclude template="../login_check.cfm">

<cfdirectory directory = "#expandpath('diffuploads')#"
name = "diffuploads"
sort="name">

<cfset db = "mysql">
<!---<cfset db = "newSPARCdb">--->
<cfset data2use="dataHF">
<cfset index="indexMaster">

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">

<html>
<head>
<title>Display Joe Diff</title>
</head>

<body>
<font color="#7f1111"  size="5" face="trebuchet ms,arial,helvetica">
Display Joe Diff</font>
<p>

<p>
<font color="#7f1111"  size="4" face="trebuchet ms,arial,helvetica">
Choose a Diff File to Display:</font>
<cfoutput>
<form action="displayJoe.cfm?db=#db#&data2use=#data2use#&index=#index#"
method="post">
</cfoutput>
    <select name="File2Use">
     <cfoutput query="DiffUploads">
     <cfif trim(name) is "." or trim(name) is "..">
     <cfelse>
     <option>#name#</option>
     </cfif>
     </cfoutput>
    </select>
<p>
<input type="submit" name="button" value="Display Set">
</form>

<hr>

<p>
<font color="#7f1111"  size="4" face="trebuchet ms,arial,helvetica">
Compare a Diff File with the Observed,PM3, and Benson Values:</font>
<form action="compareJoe.cfm" method="post">
  <select name="File2Use">
<cfoutput query="DiffUploads">
<cfif trim(name) is "." or trim(name) is "..">
<cfelse>
<option>#name#</option>
</cfif>
</cfoutput>
</select>
<p>
<input type="submit" name="button" value="Compare Set">
</form>
<hr>
WARNING CHECK THIS CODE!!!!!
<br>
Select from which database the diff file will be compared:

<br>
Select from which database the diff file will be displayed:

<p>
<font color="#7f1111"  size="4" face="trebuchet ms,arial,helvetica">
Choose a Diff File to Quasi-Display:</font>
<form action="quasiJoe.cfm" method="post">
  <select name="File2Use">
<cfoutput query="DiffUploads">
<cfif trim(name) is "." or trim(name) is "..">
<cfelse>
<option>#name#</option>
</cfif>
</cfoutput>
```

```
</select>
<p>
<input type="submit" name="button" value="Qusai-Display Set">
</form>

<p>
<a href="TS_Viewer.cfm">Home</a>
</body>
</html>
```

A.1.10 displayJoe.cfm

```
<cfinclude template="../login_check.cfm">
<cffile action="READ"
file="#expandpath('diffuploads\')##file2use#"
variable="thefile">
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<html>
<head>
<title>Joe Diff Viewer</title>
</head>
<body>
<cfoutput>
<cfset started=false>
<cfset item=0>
<cfobject action="CREATE" name="MyGifObject"
class="NewCD_Proj.clsCD" type="COM">
<cfset MyGifObject.StorageSubDirectory = "gif">
<cfobject action="CREATE" name="MyObject"
class="NewCD_Proj.SmilesCF" type="COM">
<cfset pos=0>
<cfset totalerror=0>
<cfset errorsum=0>
<cfset errorcnt=0>
<table>
<cfset xx=0>
<cfloop list="#thefile#" delimiters="#chr(13)##chr(10)#" index="line">
<cfset xx = xx+1>
<cfset item=item+1>
<cfswitch expression="#item#">
<cfcase value=1>
<cfset pos=find('** Proc',line)>
<cfif pos gt 0>
```

```
<cfset item1=line>
<cfelse>
<cfset item=item-1>
</cfif>
</cfcase>
<cfcase value=2>
<cfset item2=line>
<!--- finding the smiles--->
<cfset pos=find(":",item2)>
<cfset smiles=trim(mid(item2,pos+1,len(item2)-pos))>
<cfset smilesin='#smiles#'>
<cfset myerror=0>
<cfset myerrormessage=" ">
<cfset uniquesmiles=MyObject.GetUnique(smilesin,"myerror","myerrormessage")>
<cfif myerror>
 <cfset uniquesmiles = smiles>
</cfif>
</cfcase>
<cfcase value=3>
<cfset item3=line>
</cfcase>
<cfcase value=4>
<cfset item4=trim(line)>
</cfcase>
<cfcase value=5>
<cfset item5=trim(line)>
</cfcase>
<cfcase value=6>
<cfset item6=trim(line)>
<!--- finding the obs val --->
<cfset dpos = find("[",item6)>
<cfset dpos1 = find("/",item6,dpos)>
<cfset obs = mid(item6,dpos+1,dpos1-(dpos+1))>
</cfcase>
<cfcase value=7>
<cfset item7=trim(line)>
</cfcase>
<cfcase value=8>
<cfset item8=trim(line)>
<!--- getting the cas from the database --->
   <cfinclude template="cas_from_db.cfm">
<tr>
<td>
#item1#<br>
#item2#<br>
```

```
#item3#<br>
#item4#<br>
#item5#<br>
#item6#<br>
#item7#<br>
#item8#<br>
Steal the line(s) below:<br>
<cfif find("/",smiles) gt 0>
  hf(S).<br>
  #smiles#<br>
<cfelseif find("\",smiles) gt 0>
  hf(S).<br>
  #smiles#<br>
<cfelse>
  hf('#smiles#',S)<br>
</cfif>
CAS num: #xxx.cas#<br>

<!--- Getting the Reference and the Error--->
<cfinclude template="data_ref_error.cfm">

<cfif casnum.recordcount lt 1>
<cflocation url="error.cfm?cas=#xxx.cas#&smiles=#smiles#">
</cfif>
<cfset pos = find("http",casnum.refs)>
<cfif pos gt 0>
 <a href = #casnum.refs#>#casnum.refs#</a><br>
<cfelse>
  #casnum.refs#<br>
</cfif>
</td>
<cfif casnum.recordcount gt 0>
<cfset mycas=xxx.cas>
<cfelse>
<cfset mycas="0000">
</cfif>
<!--- found cas: see if its picture is cached --->
<cfset newdir=left(ltrim(mycas),1)> <!--- this gets subdirectory --->
<cfset temp="..\..\sparc\com_location\casgif\" &
newdir & "\" & mycas & ".gif">
<cfset testname=expandpath(temp)>
<td>
<cfif FileExists(testname)>
<!--- it was cached so use it --->
<img src="#temp#" border="0" alt=""><p>
```

```
<cfelse>
<!--- not cached so ask ChemDraw to make the gif picture and save it--->
<cfset gifname=session.sessionid & session.counter>
<cfset session.counter=session.counter+1>
<cfset MadeGIF=MyGifObject.MakeGIF("#gifname#","#smiles#")>
<cfif MadeGIF>
<cfset session.gif_file="gif\#gifname#" & ".gif">
<cfset temp="..\..\sparc\com_location\#session.gif_file#">
nc: <img src="#temp#" border="0" alt="">
<cfif mycas is "0000">
<!--- cas is 0000 don't save the pic --->
bad cas <p>
<cfelse>
<!--- real cas, save the pic --->
<cfset myfrom=expandpath(temp)>
<cfset temp="..\..\sparc\com_location\casgif\" &
newdir & "\" & mycas & ".gif">
<cfset myto=expandpath(temp)>
<cffile action="COPY" source="#myfrom#" destination="#myto#">
<p>
</cfif>
<cfelse>
Picture failed <p>
</cfif>
</cfif>
     </td>
<td>
<!---  tabulating the error --->
<cfset errorsum=casnum.error>
<cfif errorsum is "none" or
errorsum is "none reported" or
errorsum is "nr">
<cfelse>
<cfset totalerror = totalerror+(errorsum*errorsum)>
<cfset errorcnt =(errorcnt+1)>
</cfif>
</td>
</tr>
<tr><td>&nbsp;</td></tr>
<cfset item=0>
</cfcase>
</cfswitch>
</cfloop>

</table>
```

```
<cfset pos1=0>
<cfset calcrms=0>

<cfloop list="#thefile#" delimiters="#chr(13)##chr(10)#" index="line">
<cfset pos1=find('RMS',line)>
<cfif pos1 gt 0>
<cfset rms=line>
    #rms#<br>
<cfif errorcnt gt 0>
<cfset calcrms=sqr(totalerror/errorcnt)>
<cfelse>
<cfset calcrms = "none">
</cfif>
Observed RMS is #calcrms#
</cfif>
</cfloop>
</cfoutput>
<p>
<a href="TS_Viewer.cfm">Home</a>
</body>
</html>
```

A.1.11 compareJoe.cfm

```
<cfinclude template="../login_check.cfm">

<cffile action="READ"
file="#expandpath('diffuploads\')##file2use#"
variable="thefile">

<cfset db="mysql">
<cfset data2use="dataHF">
<cfset index="indexMaster">
<cfset calctable="dataCalculated">

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">

<!-------------------------------------------------------------------
This file compares the observed values vs the SPARC vs PM3 vs Benson
-------------------------------------------------------------------->
<html>
<head>
<title>Joe Diff Comparer</title>
```

```
</head>

<body>
<cfoutput>
<cfset started=false>
<cfset item=0>
<cfset pos=0>
<cfset errorPM3cnt = 0>
<cfset errorBensoncnt = 0>
<cfset totalcmpds = 0>

<cfset totalerror=0>
<cfset errorsum=0>
<cfset errorcnt=0>
<cfset ptotalerror=0>
<cfset perrorsum=0>
<cfset perrorcnt=0>
<cfset btotalerror=0>
<cfset berrorsum=0>
<cfset berrorcnt=0>



<cfobject action="CREATE" name="MyObject"
class="NewCD_Proj.SmilesCF" type="COM">
<table border=1>
<tr>
<td>Observed Values</td>
<td>SPARC</td>
<td>PM3 </td>
<td>Benson (NIST)</td>
</tr>

<cfloop list="#thefile#" delimiters="#chr(13)##chr(10)#" index="line">
<cfset item=item+1>
<cfswitch expression="#item#">
<cfcase value=1>
<cfset pos=find('** Proc',line)>
<cfif pos gt 0>
<cfset item1=line>
<cfelse>
<cfset item=item-1>
</cfif>
</cfcase>
<cfcase value=2>
```

```
<cfset item2=line>
<!---finding the smiles string --->
<cfset pos=find(":",item2)>
<cfset smiles=trim(mid(item2,pos+1,len(item2)-pos))>
<cfset smilesin='#smiles#'>
<cfset myerror=0>
<cfset myerrormessage=" ">
<cfset uniquesmiles=MyObject.GetUnique(smilesin,"myerror","myerrormessage")>
<cfif myerror>
 <!--- SMILES ERROR:  #smiles#  --->
<cfset uniquesmiles = smiles>
</cfif>
</cfcase>
<cfcase value=3>
<cfset item3=line>
</cfcase>
<cfcase value=4>
<cfset item4=trim(line)>
</cfcase>
<cfcase value=5>
<cfset item5=trim(line)>
</cfcase>
<cfcase value=6>
<cfset item6=trim(line)>
<!---finding the observed value --->
<cfset pos=find('[',item6)>
<cfset pos2=find('/',item6)>
<cfset obs=trim(mid(item6,pos+1,pos2-1-pos))>
</cfcase>
<cfcase value=7>
<cfset item7=trim(line)>
<!---finding the calculated value --->
<cfset pos7=find(':',item7)>
<cfset calc=trim(mid(item7,pos7+2,15))>
</cfcase>
<cfcase value=8>
<cfset item8=trim(line)>

<!--- getting the cas from the database --->
   <cfinclude template="cas_from_db.cfm">

<!---finding the calculated value for PM3 and Benson--->
<cfquery datasource="#db#" name="obsval">
select pm3,benson from #calctable#
where cas = #xxx.cas#
```

```
</cfquery>

<cfif obsval.recordcount gt 0>
<cfset pm3=trim(obsval.pm3)>
<cfset benson=trim(obsval.benson)>
<cfelse>
<cfset pm3="">
<cfset benson="">
</cfif>
<tr>
<td>#obs#</td>
<td>#calc# <cfset totalcmpds = totalcmpds + 1></td>
<td><cfif pm3 is "bad mol file" or
 pm3 is "error" or
 pm3 is "file not found">
 <cfset pm3 = "">
 <cfset errorPM3cnt = errorPM3cnt + 1>
</cfif>
#pm3#
</td>
<td><cfif benson is "Calculations failed">
 <cfset benson = "">
 <cfset errorBensoncnt = errorBensoncnt + 1>
</cfif>
#benson#
</td>
</tr>
<!---  tabulating the error --->
<cfinclude template="data_ref_error.cfm">

<!---  Observed error --->
  <cfset errorsum=casnum.error>
     <cfif errorsum is "none" or
     errorsum is "none reported" or
     errorsum is "nr">
     <cfelse>
     <cfset totalerror = totalerror+(errorsum*errorsum)>
<cfset errorcnt =(errorcnt+1)>
     </cfif>
<!---  PM3 error --->
<cfif pm3 neq "">
  <cfset perrorsum = abs(pm3-obs)>
       <cfset ptotalerror = ptotalerror+(perrorsum*perrorsum)>
  <cfset perrorcnt =(perrorcnt+1)>
</cfif>
```

```
<!---  Benson error --->
<cfif benson neq "">
  <cfset berrorsum = abs(benson-obs)>
       <cfset btotalerror = btotalerror+(berrorsum*berrorsum)>
  <cfset berrorcnt =(berrorcnt+1)>
</cfif>
<cfset item=0>
</cfcase>
</cfswitch>
</cfloop>
</table>

<cfset pos1=0>
<cfset rms=0>
<cfset pcalcrms=0>
<cfset bcalcrms=0>

<cfloop list="#thefile#" delimiters="#chr(13)##chr(10)#" index="line">
<cfset pos1=find('RMS',line)>
<cfif pos1 gt 0>
<cfset scalcrms=trim(mid(line,pos1+7,4))>
<cfif errorcnt gt 0>
<cfset rms=sqr(totalerror/errorcnt)>
<cfelse>
<cfset rms = "none">
</cfif>
<cfif perrorcnt gt 0>
<cfset pcalcrms=sqr(ptotalerror/perrorcnt)>
<cfelse>
<cfset pcalcrms = "none">
</cfif>
<cfif berrorcnt gt 0>
<cfset bcalcrms=sqr(btotalerror/berrorcnt)>
<cfelse>
<cfset bcalcrms = "none">
</cfif>

</cfif>
</cfloop>
<br>
<table border=1>
<tr><td>Obs RMS</td><td>SPARC RMS</td><td>PM3 RMS</td>
<td>Benson RMS</td></tr>
<tr><td>#rms#</td><td>#scalcrms#</td><td>#pcalcrms#</td>
```

```
<td>#bcalcrms#</td></tr>
</table>
<br>
<cfset totalPM3 = totalcmpds-errorPM3cnt>
<cfset totalbenson = totalcmpds-errorBensoncnt>
<table border=1>
<tr><td>Total Compounds</td><td>PM3 cmpds</td><td>Benson cmpds</td></tr>
<tr><td>#totalcmpds#</td><td>#totalPM3#</td><td>#totalbenson#</td></tr>
</table>


</cfoutput>
<p>
Copy the top tables values and paste into Excel,
then do an XY Scatter to check their linearity<br>
Set the size of the RMS, R^2 box, the X and Y axis labels,
and the tic marks to be 16.<br>
Also set the size of the data series should be 8
with fore and background colors black.<br>
Have the X and Y axis' cross so that they are on
the left and bottom of the rectangle.<br>
<br>
The second table shows the RMS of the various methods<br>
<br>
The third table can be used to show how many
compounds each method can do.<br>
<br>

<p>
<a href="TS_Viewer.cfm">Home</a>
</body>
</html>
```

A.1.12 btrain.cfm

```
<cfset p = "c:\inetpub\wwwroot\testme\">

<cfset header = "Get number of compounds from bottom of file" &
chr(13)&chr(10) & chr(13)&chr(10) &
         "%%%Parameters:" & chr(13)&chr(10) &
 "[" & chr(13)&chr(10) &
 chr(13)&chr(10) &
 "  %%  Put Parameters here"  & chr(13)&chr(10) &
 chr(13)&chr(10) &
 "]."  & chr(13)&chr(10)>

<!--- training file piece --->

<cfset header2 = session.header2>
<cfset temp = session.header4>
<!--- data --->
<!--- smiles --->
<cfset header3 = session.header3>

<!--- training file --->
<cfset tf = p & "traininX.sam">

<!--- removing and creating the files --->
<cfif fileexists(tf)>
<cffile file="#tf#" action="delete">
</cfif>
<!--- chains --->
<cfset type = "ane">
<cfinclude template= "remove_files.cfm">
<cfset type = "ene">
<cfinclude template= "remove_files.cfm">
<cfset type = "yne">
<cfinclude template= "remove_files.cfm">
<cfset type = "yene">
<cfinclude template= "remove_files.cfm">
<!--- rings --->
<cfset type = "rane">
<cfinclude template= "remove_files.cfm">
<cfset type = "rene">
<cfinclude template= "remove_files.cfm">
<cfset type = "ryne">
<cfinclude template= "remove_files.cfm">
<cfset type = "ryene">
<cfinclude template= "remove_files.cfm">
<!--- 2rings --->
<cfset type = "r2ane">
<cfinclude template= "remove_files.cfm">
<cfset type = "r2ene">
<cfinclude template= "remove_files.cfm">
<cfset type = "r2yne">
<cfinclude template= "remove_files.cfm">
<cfset type = "r2yene">
<cfinclude template= "remove_files.cfm">
```

```
<!--- 3rings --->
<cfset type = "r3ane">
<cfinclude template= "remove_files.cfm">
<cfset type = "r3ene">
<cfinclude template= "remove_files.cfm">
<cfset type = "r3yne">
<cfinclude template= "remove_files.cfm">
<cfset type = "r3yene">
<cfinclude template= "remove_files.cfm">
<!--- 4rings --->
<cfset type = "r4ane">
<cfinclude template= "remove_files.cfm">
<cfset type = "r4ene">
<cfinclude template= "remove_files.cfm">
<cfset type = "r4yne">
<cfinclude template= "remove_files.cfm">
<cfset type = "r4yene">
<cfinclude template= "remove_files.cfm">
<!--- aroms --->
<cfset type = "aane">
<cfinclude template= "remove_files.cfm">
<cfset type = "aene">
<cfinclude template= "remove_files.cfm">
<cfset type = "ayne">
<cfinclude template= "remove_files.cfm">
<cfset type = "ayene">
<cfinclude template= "remove_files.cfm">
<!--- 2aroms --->
<cfset type = "a2ane">
<cfinclude template= "remove_files.cfm">
<cfset type = "a2ene">
<cfinclude template= "remove_files.cfm">
<cfset type = "a2yne">
<cfinclude template= "remove_files.cfm">
<cfset type = "a2yene">
<cfinclude template= "remove_files.cfm">
<!--- 3aroms --->
<cfset type = "a3ane">
<cfinclude template= "remove_files.cfm">
<cfset type = "a3ene">
<cfinclude template= "remove_files.cfm">
<cfset type = "a3yne">
<cfinclude template= "remove_files.cfm">
<cfset type = "a3yene">
<cfinclude template= "remove_files.cfm">
```

```
<!--- 4aroms --->
<cfset type = "a4ane">
<cfinclude template= "remove_files.cfm">
<cfset type = "a4ene">
<cfinclude template= "remove_files.cfm">
<cfset type = "a4yne">
<cfinclude template= "remove_files.cfm">
<cfset type = "a4yene">
<cfinclude template= "remove_files.cfm">

<cfoutput>

<cfset j = 0>
<cfset k = 0>

<!--- adding the header --->
<cffile file="#tf#" action="APPEND" output="#header#">
<!---  #header#<br> --->
<cfloop query="results">
     <!--- Stuff in smiles --->
     <cfset ring=find("1",#smiles#)>
<cfset ring2=find("2",#smiles#)>
<cfset ring3=find("3",#smiles#)>
<cfset ring4=find("4",#smiles#)>

<cfset arom=find("c",#smiles#)>

     <cfset ene=find("=",#smiles#)>
     <cfset yne=find(chr(35),#smiles#)>

    <cfif ring gt 0>
<!--- aromatic rings --->
<cfif arom gt 0>
<cfif ring4 gt 0>
     <cfif yne gt 0>
        <cfif ene gt 0>
            <!--- aromatic ring yene--->
            <cfset type = "a4yene">
        <cfinclude template= "add_data.cfm">
        <cfelse>
            <!--- aromatic ring yne --->
            <cfset type = "a4yne">
        <cfinclude template= "add_data.cfm">
        </cfif>
     <cfelse>
```

```
        <cfif ene gt 0>
            <!--- aromatic ring ene --->
            <cfset type = "a4ene">
          <cfinclude template= "add_data.cfm">
        <cfelse>
            <!--- aromatic ring --->
            <cfset type = "a4ane">
        <cfinclude template= "add_data.cfm">
        </cfif>
     </cfif>
<cfelseif ring3 gt 0>
     <cfif yne gt 0>
        <cfif ene gt 0>
            <!--- aromatic ring yene--->
            <cfset type = "a3yene">
        <cfinclude template= "add_data.cfm">
        <cfelse>
            <!--- aromatic ring yne --->
            <cfset type = "a3yne">
        <cfinclude template= "add_data.cfm">
        </cfif>
     <cfelse>
        <cfif ene gt 0>
            <!--- aromatic ring ene --->
            <cfset type = "a3ene">
          <cfinclude template= "add_data.cfm">
        <cfelse>
            <!--- aromatic ring --->
            <cfset type = "a3ane">
        <cfinclude template= "add_data.cfm">
        </cfif>
     </cfif>
<cfelseif ring2 gt 0>
     <cfif yne gt 0>
        <cfif ene gt 0>
            <!--- aromatic ring yene--->
            <cfset type = "a2yene">
        <cfinclude template= "add_data.cfm">
        <cfelse>
            <!--- aromatic ring yne --->
            <cfset type = "a2yne">
        <cfinclude template= "add_data.cfm">
        </cfif>
     <cfelse>
        <cfif ene gt 0>
```

```
            <!--- aromatic ring ene --->
            <cfset type = "a2ene">
          <cfinclude template= "add_data.cfm">
        <cfelse>
            <!--- aromatic ring --->
            <cfset type = "a2ane">
        <cfinclude template= "add_data.cfm">
        </cfif>
     </cfif>
<cfelse>
     <cfif yne gt 0>
        <cfif ene gt 0>
            <!--- aromatic ring yene--->
            <cfset type = "ayene">
        <cfinclude template= "add_data.cfm">
        <cfelse>
            <!--- aromatic ring yne --->
            <cfset type = "ayne">
        <cfinclude template= "add_data.cfm">
        </cfif>
     <cfelse>
        <cfif ene gt 0>
            <!--- aromatic ring ene --->
            <cfset type = "aene">
          <cfinclude template= "add_data.cfm">
        <cfelse>
            <!--- aromatic ring --->
            <cfset type = "aane">
        <cfinclude template= "add_data.cfm">
        </cfif>
     </cfif>
</cfif>
<!--- regular rings --->
<cfelse>
 <cfif ring4 gt 0>
     <cfif yne gt 0>
      <cfif ene gt 0>
         <!--- ring yenes --->
         <cfset type = "r4yene">
       <cfinclude template= "add_data.cfm">
      <cfelse>
       <!--- ring ynes --->
           <cfset type = "r4yne">
       <cfinclude template= "add_data.cfm">
      </cfif>
     <cfelse>
      <cfif ene gt 0>
         <!--- ring enes --->
         <cfset type = "r4ene">
       <cfinclude template= "add_data.cfm">
      <cfelse>
       <!--- ring anes --->
     <cfset type = "r4ane">
     <cfinclude template= "add_data.cfm">
        </cfif>
     </cfif>
<cfelseif ring3 gt 0>
<cfif yne gt 0>
     <cfif ene gt 0>
         <!--- ring yenes --->
           <cfset type = "r3yene">
       <cfinclude template= "add_data.cfm">
     <cfelse>
       <!--- ring ynes --->
             <cfset type = "r3yne">
     <cfinclude template= "add_data.cfm">
     </cfif>
<cfelse>
     <cfif ene gt 0>
       <!--- ring enes --->
         <cfset type = "r3ene">
     <cfinclude template= "add_data.cfm">
     <cfelse>
       <!--- ring anes --->
     <cfset type = "r3ane">
     <cfinclude template= "add_data.cfm">
     </cfif>
</cfif>
<cfelseif ring2 gt 0>
<cfif yne gt 0>
     <cfif ene gt 0>
      <!--- ring yenes --->
         <cfset type = "r2yene">
     <cfinclude template= "add_data.cfm">
     <cfelse>
       <!--- ring ynes --->
         <cfset type = "r2yne">
     <cfinclude template= "add_data.cfm">
     </cfif>
<cfelse>
```

```
       <cfif ene gt 0>
         <!--- ring enes --->
         <cfset type = "r2ene">
       <cfinclude template= "add_data.cfm">
       <cfelse>
         <!--- ring anes --->
     <cfset type = "r2ane">
     <cfinclude template= "add_data.cfm">
       </cfif>
</cfif>
<cfelse>
<cfif yne gt 0>
        <cfif ene gt 0>
          <!--- ring yenes --->
          <cfset type = "ryene">
        <cfinclude template= "add_data.cfm">
        <cfelse>
         <!--- ring ynes --->
           <cfset type = "ryne">
     <cfinclude template= "add_data.cfm">
        </cfif>
<cfelse>
        <cfif ene gt 0>
         <!--- ring enes --->
           <cfset type = "rene">
         <cfinclude template= "add_data.cfm">
        <cfelse>
         <!--- ring anes --->
     <cfset type = "rane">
     <cfinclude template= "add_data.cfm">
        </cfif>
</cfif>
</cfif>
</cfif>
<!--- chains --->
<cfelse>
<cfif yne gt 0>
<cfif ene gt 0>
   <!--- ynes-enes --->
   <cfset type = "yene">
   <cfinclude template= "add_data.cfm">
<cfelse>
   <!--- ynes --->
   <cfset type = "yne">
   <cfinclude template= "add_data.cfm">
```

```
</cfif>
<cfelse>
<cfif ene gt 0>
     <!--- enes --->
          <cfset type = "ene">
<cfinclude template= "add_data.cfm">
<cfelse>
<!--- anes --->
<cfset type = "ane">
<cfinclude template= "add_data.cfm">
</cfif>
</cfif>
</cfif>
</cfloop>

<!--- Adding the number of cmpds in the file --->
<cffile file="#tf#" action = "READ" variable = "tfile">
<cfset all = "%" &j & ".">
<cfset newtfile = Replace
(tfile,
"Get number of compounds from bottom of file",
j-k & "." & Chr(13) & Chr(10) & all)>
<cffile file="#tf#" action = "WRITE" addNewLine = "No"
output = "#newtfile#">

<!--- Adds all the small files together to get a big file --->
<!--- chains --->
<cfset type = "ane">
<cfinclude template= "read_files.cfm">
<cfset type = "ene">
<cfinclude template= "read_files.cfm">
<cfset type = "yne">
<cfinclude template= "read_files.cfm">
<cfset type = "yene">
<cfinclude template= "read_files.cfm">
<!--- rings --->
<cfset type = "rane">
<cfinclude template= "read_files.cfm">
<cfset type = "rene">
<cfinclude template= "read_files.cfm">
<cfset type = "ryne">
<cfinclude template= "read_files.cfm">
<cfset type = "ryene">
<cfinclude template= "read_files.cfm">
<!--- rings2 --->
```

```
<cfset type = "r2ane">
<cfinclude template= "read_files.cfm">
<cfset type = "r2ene">
<cfinclude template= "read_files.cfm">
<cfset type = "r2yne">
<cfinclude template= "read_files.cfm">
<cfset type = "r2yene">
<cfinclude template= "read_files.cfm">
<!--- rings3 --->
<cfset type = "r3ane">
<cfinclude template= "read_files.cfm">
<cfset type = "r3ene">
<cfinclude template= "read_files.cfm">
<cfset type = "r3yne">
<cfinclude template= "read_files.cfm">
<cfset type = "r3yene">
<cfinclude template= "read_files.cfm">
<!--- rings4 --->
<cfset type = "r4ane">
<cfinclude template= "read_files.cfm">
<cfset type = "r4ene">
<cfinclude template= "read_files.cfm">
<cfset type = "r4yne">
<cfinclude template= "read_files.cfm">
<cfset type = "r4yene">
<cfinclude template= "read_files.cfm">
<!--- aromatics --->
<cfset type = "aane">
<cfinclude template= "read_files.cfm">
<cfset type = "aene">
<cfinclude template= "read_files.cfm">
<cfset type = "ayne">
<cfinclude template= "read_files.cfm">
<cfset type = "ayene">
<cfinclude template= "read_files.cfm">
<!--- 2aromatics --->
<cfset type = "a2ane">
<cfinclude template= "read_files.cfm">
<cfset type = "a2ene">
<cfinclude template= "read_files.cfm">
<cfset type = "a2yne">
<cfinclude template= "read_files.cfm">
<cfset type = "a2yene">
<cfinclude template= "read_files.cfm">
<!--- 3aromatics --->
```

```
<cfset type = "a3ane">
<cfinclude template= "read_files.cfm">
<cfset type = "a3ene">
<cfinclude template= "read_files.cfm">
<cfset type = "a3yne">
<cfinclude template= "read_files.cfm">
<cfset type = "a3yene">
<cfinclude template= "read_files.cfm">
<!--- 4aromatics --->
<cfset type = "a4ane">
<cfinclude template= "read_files.cfm">
<cfset type = "a4ene">
<cfinclude template= "read_files.cfm">
<cfset type = "a4yne">
<cfinclude template= "read_files.cfm">
<cfset type = "a4yene">
<cfinclude template= "read_files.cfm">

</cfoutput>
```

A.1.13 REMOVE_FILES.CFM

```
<cfset pp = p & type & ".sam">

<!---delete the file --->
<cfif fileexists(pp)>
<cffile file="#pp#" action="delete">
</cfif>

<cfset comment = "/****** " &  type & " ******/">

<!---create the file --->
<cffile file="#pp#" action="APPEND" output= "#comment#">
```

A.1.14 READ_FILES.CFM

```
<cfset pp = p & type & ".sam">

<cffile file="#pp#" action="READ" variable="type">
<cffile file="#tf#" action="APPEND" addNewLine="No" output="#type#">
```

A.1.15 add_data.cfm

```
<cfset fle ="c:\inetpub\wwwroot\testme\" & #type# &".sam">

<!--- get  data --->
<cfquery datasource=#application.datasource# name="hfC">
SELECT #session.select#
WHERE cas = #cas#
ORDER BY rank
</cfquery>

<cfset hf = Valuelist(hfC.prop)>
<cfif session.header3 eq "sfus.">
<cfset temp1 = hfC.temp - 273.15>
<cfset temp = temp1 & ".">
</cfif>
<cfset hf3 = Valuelist(hfC.xuse)>
<cfset hf4 = Valuelist(hfC.comments2)>

<!--- adding the data to the file--->
<cfif listlen(hf) gt 0>
    <cfset j=j+1>
<cfset lfst = ListFirst(hf3)>
<cfif lfst is "n">
<cfset k=k+1>
<cffile file="#fle#" action="APPEND"
output="/*************************************************">
     <cfif listlen(hf4) gt 0>
           <cffile file="#fle#" action="APPEND"
           output="#header2# %#listfirst(hf4)#">
     <cfelse>
        <cffile file="#fle#" action="APPEND"
        output="#header2#">
     </cfif>
        <cffile file="#fle#" action="APPEND"
        output="#temp#">
        <cfif listlen(hf) eq 1>
           <cffile file="#fle#" action="APPEND"
           output="#listfirst(hf)#.">
        <cfelse>
           <cffile file="#fle#" action="APPEND"
           output="#listfirst(hf)#. %#listrest(hf)#">
        </cfif>
        <cffile file="#fle#" action="APPEND"
        output="'#smiles#'. %#cas#">
        <cffile file="#fle#" action="APPEND"
        output="#header3#">
<cffile file="#fle#" action="APPEND"
output="***************************************************/">
<cfelse>
     <cfif listlen(hf4) gt 0>
           <cffile file="#fle#" action="APPEND"
           output="#header2# %#listfirst(hf4)#">
     <cfelse>
        <cffile file="#fle#" action="APPEND"
        output="#header2#">
     </cfif>
        <cffile file="#fle#" action="APPEND"
        output="#temp#">
        <cfif listlen(hf) eq 1>
           <cffile file="#fle#" action="APPEND"
           output="#listfirst(hf)#.">
        <cfelse>
           <cffile file="#fle#" action="APPEND"
           output="#listfirst(hf)#. %#listrest(hf)#">
        </cfif>
        <cffile file="#fle#" action="APPEND"  output="'#smiles#'. %#cas#">
        <cffile file="#fle#" action="APPEND" output="#header3#">
</cfif>
</cfif>
```

A.1.16 DATA2PARAM_TOP.CFM

```
<cfinclude template="../login_check.cfm">

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">

<html>
<head>
<title>Convert SPARC Data to Trainable Parameters</title>
</head>

<body>
Paste the SPARC data into the form, click <B>Convert</B>.

<Form action="data2params.cfm" method="post">
<P>
```

```
<Textarea name="data" rows="20" cols="60"></textarea>
<P>
<input type="submit" value="Convert">
</form>

</body>
</html>
```

A.1.17 DATA2PARAMS.CFM

```
<cfinclude template="../login_check.cfm">

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">

<html>
<head>
<title>Training File Parameters</title>
</head>

<body>
<cfoutput>
<cfset delim=chr(13) & chr(10)>
<cfset data=#form.data#>

<cfset j=0>
<cfloop list="#form.data#" delimiters="#delim#" index="a_line">
<cfset pos=find(delim,a_line)+1>
<cfset j=j+1>
<cfif pos gt 0>
  <cfset pos1 = findnocase("(",a_line)>
  <cfset pos2 = findnocase(",",a_line)>
  <cfset name= mid(a_line,pos,pos1-pos)>
  <cfset value= mid(a_line,pos1+1,pos2-pos1-1)>
  <cfset parameter= "[" & value & "," & name & ",1,1]," & delim>
  #parameter#<br>
</cfif>
</cfloop>

</cfoutput>

</body>
</html>
```

A.2 Quality Control Programs

A.2.1 Benchscript Script

```
#!/bin/sh

# define variables here
SPARCG='tad@sunlc3.chem.uga.edu, raj@sunlc3.chem.uga.edu, \
butch@sunlc3.chem.uga.edu, hilal.said@epa.gov'
##SPARCG='tad@sunlc3.chem.uga.edu'
FROM='butch@sunlc3.chem.uga.edu'
FILE_NEW='bench.out'
FILE_OLD='bench_old.out'
DIFF_FILE='diff.out'
BAD='bad.dat'

############### if script dies - send mail saying so
trap 'nail -s "Script died, something wrong" -r $FROM $SPARCG ' 1 2 3

############### copy previous days *.out to *_old.out

mv -f $FILE_NEW $FILE_OLD

############### remove bad.dat

rm -f $BAD

############### run benchsparc for each bench file, throw away output junk

#------ PROPERTY ----- LINES

#------ pKa ----- 1-2715
#/home3/trainfiles/pka/pka_jan_04.tra

#------ Hf ----- 2716-3307
#/home3/trainfiles/hf/hydrocarbons.sam

#------ Properties ----- 3308-6766
#/home3/trainfiles/bigset/save_vp-03.tra

#------ Hydrolysis - acid ----- 6767-7445
#/home3/trainfiles/hydrolysis/acide_hyde.tra

#------ Hydrolysis - base ----- 7446-8140
```

```
#/home3/trainfiles/hydrolysis/base_hyd.tra

#------ Hydrolysis - neutral ----- 8141-8309
#/home3/trainfiles/hydrolysis/gen_hyde.tra

#------ Electron Affinity ----- 8310-8562
#/home3/trainfiles/electronaffinity/electron_affinity_03.tra

#------ Diffusion ----- 8563-8672
#/home3/trainfiles/diffusion/diffuse.dat

#------ Ehalf ----- 8673-8991
#/home3/trainfiles/ehalf/ehalf_all_03.tra

#------ Gas and NonAqueous pKa ----- 8992-9799
#/home3/trainfiles/ehalf/charge_03.tra
#/home3/trainfiles/ehalf/charge_nr2n_03.tra
#/home3/trainfiles/ehalf/charge_methyl_03_10.tra

#----- Hydration ----- 9800-9864
#/home3/trainfiles/hydration/aldeketohyd36.tra
#/home3/trainfiles/hydration/quinaz_other.tra

echo Running BigBench
yes | benchsparc bench ##>& /dev/null
echo Done

############## if SPARC dies
if [ -f $BAD ]; then
    cat $BAD | nail -s 'SPARC died, something REALLY wrong' -r $FROM $SPARCG

## if it doesn't, then do rest of stuff
else
    ## perform difference on $FILE_NEW and $FILE_OLD \
    and redirect results to $DIFF_FILE
    qc_diff_exe
    mv -f largeerror.out $DIFF_FILE
    cat smallerror.out >> $DIFF_FILE

    ## if diff not 66(empty except for headers), mail results to SPARCG

    DIFF_SIZE=`/usr/bin/filesize $DIFF_FILE`

    if (test $DIFF_SIZE != 66) then
cat $DIFF_FILE | nail -s 'QC Results have changed!!' -r $FROM $SPARCG
```

```
    else
echo | nail -s 'Delete this message:  QC Results OK!' -r $FROM $SPARCG
    fi
fi
```

A.2.2 Benchsparc Script

```
#!/bin/sh
#

GOAL="benchbatch($1)."

/usr/local/bin/sicstus -l /home3/sparc3/sparc.pro
--goal $GOAL -a $HOME compile
```

A.2.3 Benchsparc predicate, part of batchmod.pro

```
/**************************************************************
 Benchmark code goes here
**************************************************************/
benchbatch(FileName) :-
        set_working_dir_list,
        get_toggle_status(l,_),
        new_pk,
        set_output(user_output),
        set_input(user_input),
        retractalls(molecule_data(_,_,_)),
        cls,

%%% INPUT FILE SECTION %%%%
        concatenate_atomlist_to_atom([FileName,'.dat'],DatFile),

%%% DRIBBLE FILE SECTION %%%%
%         Drib = '',
        open_null_stream(Dribble),
        DribX = Dribble,

%%% OUTPUT FILE SECTION %%%%
        concatenate_atomlist_to_atom([FileName,'.out'],Out),
        execute(catch(close(Out),_,fail),_),
        open(Out,write,_Output,[alias(Out)]),
        OutX = Out,
%%% MULTIPLIER FILE SECTION %%%%
        assert(do_multiply),

%%% REST OF STUFF %%%%
        open('time.dat',write,Time),
        time(X),
        write(Time,X),nl(Time),
        close(Time),

        batchmode1(DatFile,OutX,DribX),

        open('time.dat',append,Time2),
        time(Y),
        write(Time2,Y),nl(Time2),
        write(Time2,'batch'),nl(Time2),
        close(Time2),
        halt.
benchbatch(_) :-
        halt.
```

A.3 Common Area Script

A.3.1 Build_area Script

```
#!/bin/sh
#
######################################################
#  Copy this file to the area you would like to build
#  this will reach into ~sparc/user_template area and
#  get all the necessary files to set up your area,
#  create the necessary directories, etc
#
#  TSW 5-6-2004
######################################################

### creates necessary directories
mkdir mol
mkdir ss
mkdir last
mkdir last1 last2 last3 last4 last5 last6 last7 last8 last9 last10
mkdir last11 last12 last13 last14 last15 last16 last17 last18 last19 last20
```

```
### gets the exes
cp ~sparc/user_template/runsparc ./.
cp ~sparc/user_template/build_library ./.
#cp ~sparc/user_template/doreparam ./.
#cp ~sparc/user_template/trainsparc ./.
cp ~sparc/user_template/prolog.cmd ./.

### gets the files
cp ~sparc/user_template/kovtemp.dat ./.
cp ~sparc/user_template/defaults.pl ./.
cp ~sparc/user_template/current.pl ./.
cp ~sparc/user_template/multipliers.dat ./.
cp ~sparc/user_template/propunit.dat ./.
cp ~sparc/user_template/passnum.dat ./.
cp ~sparc/user_template/header.dat ./.
cp ~sparc/user_template/batch_init.pro ./.
cp ~sparc/user_template/stripped.ss ./ss/.
cp ~sparc/user_template/test.ss ./ss/.
cp ~sparc/user_template/unique.ss ./ss/.

### creates links
temp=/home3/sparc/
ln -s $temp'batstat' batstat
ln -s $temp'cas' cas
ln -s $temp'notuniq' notuniq
ln -s $temp'rebuild' rebuild
ln -s $temp'reparam' reparam
ln -s $temp'thresh' thresh

### builds special files
###
touch obs.sam
touch newparam.sam

###
for j in last last1 last2 last3 last4 last5 last6 last7
last8 last9 last10 last11 last12 last13 last14
last15 last16 last17 last18 last19 last20
do
  touch $j/newparam.sam
done

### make progs.dat

echo "emacs-nox
```

```
/home3/sparc3/" > progs.dat
echo $HOME'/' >> progs.dat
echo "First line is the default editor (path&name if needed default is AHED)
Second is the root area for SPARC
Third is home directory
" >> progs.dat

### make logicals.pl

echo "/*************************************************************
This file contains the definitions of a set of logical names which should
also appear in your login file  This allows SPARC to substitute the full
directory path whenever it has access either to the logical name for the
path or the directory name itself.  This file should be modified whenever
the logical name definitions in your login file are changed.  It is used
by the procedure load_logicals to create a set of facts which have the
form: logical_name(Name,Path,Directory).
*************************************************************/"
> logicals.pl

echo "[ %     LOGICAL NAME            FULL PATH               DIRECTORY
 [      defaultdir,             '',            'current directory.'],
 [      sparcdir,               '/home3/sparc3/',    sparc           ],
 [      pkadir,                 '/home3/sparc/',     pka             ],
 [      hydrodir,               '/home3/sparc3/',    hydro           ],"
>> logicals.pl
temp="[lightdir,'"$HOME"/','"$HOME"/'],"
echo $temp >> logicals.pl
echo " [rootdir,'/home3/sparc3/',anv],[mdata,'/home3/sparc3/',fr],"
>> logicals.pl
temp="[sdata,'"$HOME"/ss/',ss]
]."
echo $temp >> logicals.pl

### make conv.cfg
echo "Below is the path to the *.MOL files and the CONV.TAB file" > conv.cfg
temp=$HOME'/mol/'
echo $temp >> conv.cfg
echo "Next is the path to the *.SS file storage area:" >> conv.cfg
temp=$HOME'/ss/'
echo $temp >> conv.cfg
echo "Next is the path to the bild.dat file" >> conv.cfg
temp=$HOME'/'
echo $temp >> conv.cfg
echo "Next is the path to the *.DAT files" >> conv.cfg
```

```
temp='/home3/sparc/bild/'
echo $temp >> conv.cfg

### make doreparam
echo "#!/bin/sh" > doreparam
temp='cd '$HOME
echo $temp  >> doreparam
temp='sicstus --iso  -l /home3/sparc3/backgr.pro
--goal myreparam.  -a '$HOME
echo $temp  >> doreparam
chmod +x doreparam

### make trainsparc
echo "#!/bin/sh" > trainsparc
temp='cd '$HOME
echo $temp  >> trainsparc
temp='sicstus --iso  -l /home3/sparc3/backgr.pro
--goal background.  -a '$HOME
echo $temp  >> trainsparc
chmod +x trainsparc

### delete itself after run
rm build_area
```

Appendix B

Compounds Used to Develop the Model

B.1 Hydrocarbons

Table B.1: Observed and calculated values of hydrocarbons

Name	**CAS**	**Observed (kJ mol^{-1})**	**SPARC**
Ethane	74840	-83.80	-87.26
Propane	74986	-105.64	-107.46
Isobutane	75285	-134.20	-135.43
Butane, 2,2-dimethyl-	75832	-187.51	-185.66
Butane, 2-methyl-	78784	-155.19	-154.72
Butane, 2,3-dimethyl-	79298	-179.55	-178.33
Pentane, 3-methyl-	96140	-173.96	-173.84
Butane	106978	-125.60	-127.99
Pentane, 2-methyl-	107835	-176.58	-175.04
Pentane, 2,4-dimethyl-	108087	-204.31	-199.99
Pentane	109660	-148.73	-148.67
Hexane	110543	-169.59	-169.40

Continued on next page

Name	**CAS**	**Observed (kJ mol^{-1})**	**SPARC**
Octane	111659	-211.22	-210.92
Nonane	111842	-231.01	-231.69
n-Dodecane	112403	-290.27	-293.98
Eicosane	112958	-461.55	-460.10
Decane	124185	-252.46	-252.45
Heptane	142825	-191.92	-190.16
Propane, 2,2-dimethyl-	463821	-167.56	-170.33
Butane, 2,2,3-trimethyl-	464062	-205.95	-203.64
Pentane, 2,2,4-trimethyl-	540841	-225.43	-226.80
Hexadecane	544763	-378.54	-377.04
Pentane, 2,3,3-trimethyl-	560214	-217.46	-217.84
Pentane, 3,3-dimethyl-	562492	-203.33	-200.79
Hexane, 3,3-dimethyl-	563166	-221.81	-219.47
Pentane, 2,2,3-trimethyl-	564023	-221.31	-217.39
Pentane, 2,3-dimethyl-	565593	-200.68	-196.64
Pentane, 2,3,4-trimethyl-	565753	-218.96	-217.83
Hexane, 3,4-dimethyl-	583482	-214.85	-214.87
Hexane, 2,3-dimethyl-	584941	-215.86	-216.63
Hexane, 3-methyl-	589344	-195.35	-194.11
Hexane, 2,4-dimethyl-	589435	-221.70	-218.60
Heptane, 4-methyl-	589537	-214.42	-214.37
Heptane, 3-methyl-	589811	-214.96	-214.86
Pentane, 2,2-dimethyl-	590352	-208.15	-204.41
Hexane, 2,2-dimethyl-	590738	-226.70	-224.61

Continued on next page

Name	**CAS**	**Observed (kJ mol^{-1})**	**SPARC**
Hexane, 2-methyl-	591764	-198.56	-195.79
Hexane, 2,5-dimethyl-	592132	-224.78	-221.54
Heptane, 2-methyl-	592278	-218.00	-216.53
Octadecane	593453	-418.57	-418.57
Butane, 2,2,3,3-tetramethyl-	594821	-232.74	-230.13
Pentane, 3-ethyl-2-methyl-	609267	-212.99	-214.71
Pentane, 3-ethyl-	617787	-192.20	-192.79
Hexane, 3-ethyl-	619998	-213.11	-213.01
Tridecane	629505	-313.80	-314.74
Tetradecane	629594	-334.64	-335.51
Pentadecane	629629	-355.52	-356.27
Heptadecane	629787	-397.28	-397.80
Nonadecane	629925	-439.93	-439.33
Nonane, 2-methyl-	871830	-262.79	-257.81
Pentane, 3-ethyl-3-methyl-	1067089	-216.33	-215.82
Pentane, 3,3-diethyl-	1067205	-233.30	-230.93
Pentane, 3-ethyl-2,4-dimethyl-	1068877	-229.28	-234.94
Hexane, 2,3,5-trimethyl-	1069530	-244.50	-240.33
Pentane, 2,2,4,4-tetramethyl-	1070877	-243.56	-243.79
Heptane, 2,2-dimethyl-	1071267	-248.19	-245.09
Undecane	1120214	-273.82	-273.21
Pentane, 2,2,3,4-tetramethyl-	1186534	-236.90	-232.32
Hexane, 2,2,5-trimethyl-	3522949	-254.84	-250.18
Pentane, 2,2,3,3-tetramethyl-	7154792	-237.10	-239.70

Continued on next page

Name	CAS	Observed (kJ mol^{-1})	SPARC
11-n-Butyldocosane	13475768	-591.74	-586.41
Nonane, 5-methyl-	15869859	-260.79	-255.87
Hexane, 2,2,3-trimethyl-	16747254	-242.47	-235.63
Hexane, 2,2,4-trimethyl-	16747265	-244.41	-241.23
Hexane, 2,3,3-trimethyl-	16747287	-240.43	-236.20
Hexane, 2,4,4-trimethyl-	16747301	-240.99	-241.37
Hexane, 3,3,4-trimethyl-	16747312	-236.77	-231.49
Pentane, 3-ethyl-2,2-dimethyl-	16747323	-232.59	-230.53
Pentane, 2,3,3,4-tetramethyl-	16747389	-236.10	-233.11
Docosane, 5-butyl-	55282161	-588.78	-585.97
Heneicosane, 11-decyl-	55320064	-702.99	-690.71
1,1,2,2-Tetra-t-butylethane	62850219	-263.86	-273.05
1,3-Butadiene, 2-methyl-	78795	75.70	80.94
1-Butene	106989	-0.63	-2.31
1,3-Butadiene	106990	111.11	112.38
1-Pentene, 2,4,4-trimethyl-	107391	-107.59	-112.51
2-Pentene, 2,4,4-trimethyl-	107404	-106.85	-110.75
1-Pentene	109671	-22.73	-23.27
1-Octene	111660	-83.06	-85.47
1-Dodecene	112414	-168.79	-168.53
Propene	115071	20.41	18.62
1-Propene, 2-methyl-	115117	-17.90	-17.17
2-Butene, 2-methyl-	513359	-43.38	-43.43
1,3-Butadiene, 2,3-dimethyl-	513815	45.10	49.05

Continued on next page

Name	CAS	Observed (kJ mol^{-1})	SPARC
1-Butene, 3,3-dimethyl-	558372	-61.84	-60.30
1-Butene, 3-methyl-	563451	-28.87	-29.19
1-Butene, 2-methyl-	563462	-36.94	-37.62
1-Butene, 2,3-dimethyl-	563780	-65.18	-63.41
2-Butene, 2,3-dimethyl-	563791	-73.26	-70.75
2-Butene, (Z)-	590181	-7.70	-9.30
1,4-Pentadiene	591935	106.30	102.78
1-Hexene	592416	-43.25	-44.25
1,5-Hexadiene	592427	82.71	81.94
1-Heptene	592767	-63.84	-65.10
1-Butene, 2,3,3-trimethyl-	594569	-86.89	-91.55
2-Pentene, 3-methyl-, (E)-	616126	-65.60	-63.79
2-Butene, (E)-	624646	-10.80	-14.78
2-Pentene, 2-methyl-	625274	-66.90	-64.02
2-Pentene, 2,4-dimethyl-	625650	-90.90	-88.44
2-Pentene, (Z)-	627203	-28.58	-29.98
1-Hexadecene	629732	-253.10	-251.59
2-Pentene, (E)-	646048	-33.67	-35.68
2-Pentene, 4-methyl-, (E)-	674760	-63.83	-61.95
2-Pentene, 4,4-dimethyl-, (E)-	690084	-91.15	-90.81
3-Hexene, 2,2-dimethyl-, (Z)-	690926	-100.91	-105.17
3-Hexene, 2,2-dimethyl-, (E)-	690937	-109.97	-111.71
1-Pentene, 4-methyl-	691372	-50.75	-49.57
2-Pentene, 4-methyl-, (Z)-	691383	-58.98	-56.02

Continued on next page

Name	CAS	Observed (kJ mol^{-1})	SPARC
3-Hexene, 2,5-dimethyl-, (E)-	692706	-124.27	-109.13
1-Pentene, 3-methyl-	760203	-50.73	-48.87
Pentane, 3-methylene-	760214	-58.46	-58.04
1-Pentene, 4,4-dimethyl-	762629	-81.60	-80.66
2-Pentene, 4,4-dimethyl-, (Z)-	762630	-74.22	-84.53
1-Pentene, 2-methyl-	763291	-61.35	-58.38
1,3,5-Hexatriene, (E)-	821078	168.00	171.53
1-Decene	872059	-125.50	-127.00
2-Pentene, 3-methyl-, (Z)-	922623	-64.76	-63.56
1,3-Pentadiene, (Z)-	1574410	82.72	84.49
1,3-Pentadiene, (E)-	2004708	75.77	78.76
1-Pentene, 2,4-dimethyl-	2213323	-85.49	-84.14
1,3,5-Hexatriene, (Z)-	2612466	172.00	177.54
1,3-Butadiene, 2-ethyl-	3404635	63.60	60.68
1-Hexene, 5-methyl-	3524730	-67.52	-70.40
3-Hexene, 3-methyl-, (E)-	3899363	-78.89	-84.38
2-Hexene, (E)-	4050457	-53.92	-56.63
3-Hexene, 3-methyl-, (Z)-	4914890	-82.27	-84.15
2,4-Hexadiene, (E,Z)-	5194503	48.00	51.01
2,4-Hexadiene, (E,E)-	5194514	44.00	45.13
(Z),(Z)-2,4-Hexadiene	6108618	52.00	56.88
(Z)-2-Heptene	6443921	-70.93	-71.78
1,4-Hexadiene, (Z)-	7318674	77.00	74.91
trans-1,4-Hexadiene	7319008	74.00	69.19

Continued on next page

Name	**CAS**	**Observed (kJ mol^{-1})**	**SPARC**
1-Butene, 2-ethyl-3-methyl-	7357939	-81.45	-83.82
3-Hexene, (Z)-	7642093	-48.75	-50.64
(Z)-3-Heptene	7642106	-70.35	-71.50
2-Hexene, (Z)-	7688213	-50.61	-50.84
(Z)-2,5-Dimethyl-3-hexene	10557445	-116.26	-102.70
3-Hexene, (E)-	13269528	-52.96	-56.58
1,3-Hexadiene, (Z)-	14596920	59.00	63.83
2-Heptene, (E)-	14686136	-75.41	-77.61
3-Heptene, (E)-	14686147	-74.56	-77.54
1-Pentene, 3-ethyl-2-methyl-	19780666	-101.70	-101.21
(E)-1,3-Hexadiene	20237347	54.00	57.86
Propyne	74997	185.40	191.12
1-Butyne	107006	165.20	167.83
1,3-Butadiyne	460128	464.00	469.50
2-Butyne	503173	145.10	148.94
1-Butyne, 3-methyl-	598232	136.40	136.78
1-Pentyne	627190	144.30	146.04
2-Pentyne	627214	128.90	125.64
1-Heptyne	628717	103.80	104.03
1-Octyne	629050	80.70	83.22
1-Hexyne	693027	122.30	124.92
2-Hexyne	764352	107.70	103.83
1-Decyne	764932	41.90	41.67
1-Butyne, 3,3-dimethyl-	917920	104.52	97.45

Continued on next page

Name	CAS	Observed (kJ mol^{-1})	SPARC
3-Hexyne	928494	105.40	102.18
2-Heptyne	1119659	84.80	82.70
4-Octyne	1942456	60.10	58.45
5-Decyne	1942467	18.70	16.12
2-Decyne	2384705	23.60	20.22
3-Decyne	2384852	21.80	17.46
4-Decyne	2384863	19.90	16.38
3-Heptyne	2586892	82.80	80.33
2-Octyne	2809678	63.80	61.80
2,4-Hexadiyne	2809690	374.62	379.26
1-Nonyne	3452093	62.30	62.44
3-Octyne	15232765	62.50	59.17
2-Nonyne	19447291	43.60	41.00
3-Nonyne	20184898	42.00	38.27
4-Nonyne	20184912	42.00	37.29
1-Buten-3-yne	689974	295.00	293.09
3-Penten-1-yne, (Z)-	1574409	258.00	263.35
3-Penten-1-yne, (E)-	2004695	259.00	257.87
(Z)-Hexa-1,5-diyne-3-ene	16668670	541.80	535.79
(E)-Hexa-1,5-diyne-3-ene	16668681	538.10	530.31
Cyclopropane	75194	53.30	49.99
Cyclopentane, methyl-	96377	-108.28	-108.55
Cyclohexane, methyl-	108872	-154.78	-153.57
Cyclohexane	110827	-124.96	-125.54

Continued on next page

Name	**CAS**	**Observed (kJ mol^{-1})**	**SPARC**
Cyclobutane	287230	24.01	19.14
Cyclopentane	287923	-78.85	-80.25
Cycloheptane	291645	-119.24	-124.86
Cyclooctane	292648	-126.04	-127.59
Cyclononane	293550	-134.33	-140.67
Cyclodecane	293969	-154.15	-158.62
Cycloundecane	294417	-177.41	-177.56
Cyclotridecane	295023	-244.23	-244.29
Cyclopentadecane	295487	-301.09	-299.24
Cyclohexadecane	295658	-323.89	-324.17
Cycloheptadecane	295976	-346.11	-348.00
Cyclohexane, 1,1-dimethyl-	590669	-181.16	-182.69
Methylcyclopropane	594116	22.43	20.84
Cyclohexane, 1,4-dimethyl-, cis-	624293	-176.80	-174.53
Cyclohexane, 1,3-dimethyl-, cis-	638040	-183.47	-181.61
Cyclopentane, 1,2-dimethyl-, trans-	822504	-138.38	-136.05
Cyclopropane, 1,2-dimethyl-, cis-	930187	-2.19	-4.13
Cyclopentane, 1-ethyl-2-methyl-, cis-	930892	-153.25	-151.98
trans-1-Ethyl-2-methyl-cyclopentane	930905	-157.55	-155.90
Cyclopropane, ethyl-	1191964	0.54	0.29
Cyclopentane, 1,2-dimethyl-, cis-	1192183	-132.48	-132.13
Cyclopropane, 1,1-dimethyl-	1630940	-10.74	-11.65
Cyclopentane, 1,1-dimethyl-	1638262	-140.33	-138.69
Cyclopentane, ethyl-	1640897	-130.87	-128.72

Continued on next page

Name	**CAS**	**Observed (kJ mol^{-1})**	**SPARC**
Cyclohexane, ethyl-	1678917	-172.56	-173.73
Cyclohexane, propyl-	1678928	-193.13	-194.39
Cyclohexane, butyl-	1678939	-213.25	-215.27
Cyclopentane, 1,3-dimethyl-, trans-	1759586	-136.24	-136.86
Cyclohexane, decyl-	1795160	-339.10	-339.72
Dodecylcyclohexane	1795171	-377.65	-381.25
Cyclopentane, decyl-	1795217	-294.54	-294.66
Cyclohexane, 1,3,5-trimethyl-, (1α,3α,5α)-cis	1795273	-212.09	-209.65
Cyclopentane, propyl-	2040962	-149.03	-149.39
1,1,4-Trimethylcycloheptane	2158556	-211.37	-210.06
Cyclohexane, 1,2-dimethyl-, cis-	2207014	-174.35	-173.72
Cyclohexane, 1,3-dimethyl-, trans-	2207036	-176.27	-174.53
Cyclohexane, 1,4-dimethyl-, trans-	2207047	-183.60	-181.61
Cyclopropane, 1,2-dimethyl-, trans-	2402064	-6.59	-8.05
Cyclopentane, 1,3-dimethyl-, cis-	2532583	-134.24	-136.86
Cyclopentane, 1-ethyl-3-methyl-, trans-	2613652	-157.39	-157.03
Cyclopentane, 1-ethyl-3-methyl-, cis-	2613663	-155.78	-157.03
Eicosane, 3-cyclohexyl-	4443576	-539.40	-546.80
Eicosane, 9-cyclohexyl-	4443612	-546.63	-546.64
Cyclobutane, ethyl-	4806615	-30.32	-29.92
Cyclohexane, 1-ethyl-2-methyl-, cis-	4923777	-193.19	-193.55
Cyclohexane, 1-ethyl-2-methyl-, trans-	4923788	-197.26	-200.63
Cyclohexane, 1-ethyl-4-methyl-, cis-	4926787	-195.54	-194.68

Continued on next page

Name	CAS	Observed (kJ mol^{-1})	SPARC
Cyclohexane, 1-ethyl-1-methyl-	4926903	-198.21	-199.44
Heptylcyclohexane	5617414	-288.33	-277.43
Cyclohexane, 1-ethyl-4-methyl-, trans-	6236880	-202.94	-201.77
Pentacosane, 13-cyclohexyl-	6697150	-638.99	-650.46
Cyclopentane, heneicosyl-	6703828	-525.20	-523.08
Cyclohexane, (1-decylundecyl)-	6703997	-557.55	-567.40
Cyclohexane, 1,2-dimethyl-, trans-	6876239	-179.85	-180.80
1,1,4,4-Tetramethyl-cyclodecane	15841119	-275.89	-272.94
1,1,5,5-Tetramethyl-cyclodecane	16723890	-266.39	-272.94
Cyclopentane, 1-ethyl-1-methyl-	16747505	-156.70	-155.45
cis-1-Ethyl-3-methyl-cyclohexane	19489102	-202.60	-201.77
1,1-Dimethyl-2-ethyl-cyclopropane	41845470	-58.95	-57.64
1,1-Dimethyl-2-propyl-cyclopropane	41845481	-80.29	-77.46
1,1-Dimethyl-2-hexylcyclopropane	41845492	-142.69	-139.03
trans-1,2-Diethyl-cyclopropane	71032661	-50.05	-47.87
cis-1,2-Diethylcyclopropane	71032672	-47.03	-43.95
Terpilene	99865	-20.60	-14.39
Cyclohexene, 4-ethenyl-	100403	69.11	66.76
Cyclohexene	110838	-5.89	-10.50
Cyclopentene	142290	30.45	33.97
Fulvene	497201	221.72	225.99
Cyclohexene, 4-methyl-1-(1-methylethyl)-	500005	-109.12	-115.40
1,3-Cyclopentadiene	542927	134.96	134.27
1,3,5-Cycloheptatriene	544252	187.00	190.69

Continued on next page

Name	CAS	Observed (kJ mol^{-1})	SPARC
Cyclohexene, 4-methyl-	591479	-39.89	-38.54
Cyclohexene, 1-methyl-	591491	-44.66	-42.98
1,3-Cyclohexadiene	592574	101.38	93.31
1,4-Cyclohexadiene	628411	101.54	104.53
Cycloheptene	628922	-9.16	-11.95
1,3,5,7-Cyclooctatetraene	629209	293.04	284.56
1,5,9-Cyclododecatriene, (E,E,E)-	676222	87.26	88.85
Ethenylcyclopropane	693867	128.30	126.39
Cyclopentene, 1-methyl-	693890	-7.45	-6.55
Cyclopentene, 3-ethyl-	694359	-16.37	-14.58
Cyclohexane, ethenyl-	695125	-47.87	-48.27
cis,cis,cis-Cyclononatriene	696866	189.00	189.02
1,5,9-Cyclododecatriene, (E,E,Z)-	706310	90.56	88.85
Cyclopentene, 1,2-dimethyl-	765479	-41.40	-41.58
Cyclobutene	822355	157.00	160.50
3-Methylenecyclopentene	930267	112.72	110.55
cis-Cyclooctene	931873	-22.70	-13.17
trans-Cyclooctene	931895	20.00	34.74
Cyclohexane, ethylidene-	1003641	-62.48	-66.17
Cyclobutane, methylene-	1120565	118.44	118.43
Cyclopentene, 3-methyl-	1120623	5.44	5.62
Cyclohexane, methylene-	1192376	-25.20	-29.41
Cyclohexene, 1-ethyl-	1453243	-65.45	-62.84
Cyclopentane, methylene-	1528309	10.35	10.30

Continued on next page

Name	CAS	Observed (kJ mol^{-1})	SPARC
1,5-Cyclooctadiene, (Z,Z)-	1552121	98.76	101.25
Cyclopentene, 4-methyl-	1759815	11.71	5.66
Ethylidenecyclopentane	2146374	-21.26	-26.49
Cyclopentene, 1-ethyl-	2146385	-22.92	-26.75
cis-2-Methyl-1-vinylcyclopropane	2628571	103.00	102.10
Cyclopropene	2781853	277.10	278.27
1-Methylcyclopropene	3100047	243.57	246.86
Cyclopropane, tris(methylene)-	3227905	396.00	392.67
Cyclopentane, 2-propenyl-	3524752	-26.34	-23.96
1,3,6-Cyclooctatriene	3725302	196.30	204.45
Cyclopentane, ethenyl-	3742345	0.41	-3.14
cis,cis-1,3-Cyclooctadiene	3806595	84.10	90.03
(Z,E)-1,3-Cyclooctadiene	3806608	146.00	137.94
6-Methylfulvene	3839507	181.13	189.20
1,3-Cycloheptadiene	4054380	93.90	89.75
1,3-Cyclopentadiene, 5,5-dimethyl-	4125182	86.60	75.67
α-Phellandrene	4221981	-13.79	-8.32
Cyclopropane, (1-methylethenyl)-	4663223	89.23	91.24
Cyclohexane, 1,4-bis(methylene)-	4982201	64.39	66.72
1,5-Cyclooctadiene, (E,Z)-	5259712	158.00	149.16
Cyclohexene, 1-methyl-4-(1-methylethyl)-	5502885	-111.50	-116.03
Methylenecyclopropane	6142730	201.00	193.80
trans-1,2-Divinylcyclobutane	6553486	177.00	174.21
cis-1,2-Divinylcyclobutane	16177461	188.00	185.21

Continued on next page

Name	CAS	Observed (kJ mol^{-1})	SPARC
Cyclopropane, 1,1-diethenyl-	17085846	202.00	200.82
1,5-Cyclooctadiene, (E,E)-	17612509	196.00	197.08
Cyclopentane,1,2-bis(methylene)-	20968701	92.62	89.49
3-(cis-Ethylidene)-1-cyclopentene	22704387	80.13	73.75
Cyclopropylacetylene	6746947	287.74	296.21
Cyclopropane, 1,1-diethynyl-	72323661	541.04	533.92
1-Cyclopropylpenta-1,3-diyne	116316768	480.28	484.45
1,1'-Bicyclohexyl	92513	-215.14	-218.29
Bicyclo[1.1.0]butane	157335	217.00	217.00
Spiropentane	157404	185.10	158.03
Spiro(4,4)nonane	175939	-99.14	-100.33
Spiro(4-5)decane	176636	-149.40	-144.52
Spiro[5.5]undecane	180438	-188.25	-188.70
Spiro[5.6]dodecane	181157	-192.53	-188.02
Bicyclo[2.1.0]pentane	185944	155.88	165.17
Bicyclo[2.2.0]hexane	186049	123.15	116.30
Bicyclo[4.2.0]octane	278308	-27.56	-28.00
Norbornane	279232	-52.35	-52.22
Bicyclo[2.2.2]octane	280331	-98.90	-96.95
Bicyclo[3.3.1]nonane	280659	-124.91	-124.88
Bicyclo[3.3.2]decane	283501	-113.77	-113.76
Bicyclo[3.1.0]hexane	285585	43.14	41.75
Bicyclo[4.1.0]heptane	286088	4.36	1.57
Naphthalene, decahydro-, cis-	493016	-166.93	-169.37

Continued on next page

Name	**CAS**	**Observed (kJ mol^{-1})**	**SPARC**
Naphthalene, decahydro-, trans-	493027	-178.04	-187.75
Endo-2-methylnorbornane	765902	-81.27	-78.44
Bicyclo[2.2.1]heptane, 2-methyl-, exo-	872786	-81.67	-82.36
Cyclopentylcyclohexane	1606082	-173.28	-173.37
1,1’-Bicyclopentyl	1636391	-127.98	-128.44
Pentalene, octahydro-, cis-	1755051	-94.75	-92.80
7,7-Dimethyl-bicyclo[2.2.1]heptane	2034539	-102.07	-107.77
Bicyclo[2.2.1]heptane, 2-ethyl-	2146410	-94.72	-102.18
1-Methylbicyclo[4.1.0]-heptane	2439794	-17.38	-20.82
cis-4a-Methyl-decahydronaphthalene	2547264	-188.86	-186.92
trans-4a-Methyl-decahydronaphthalene	2547275	-194.57	-205.29
Dicyclopentylmethane	2619343	-148.95	-149.32
1H-Indene, octahydro-, trans-	3296502	-129.35	-122.79
1H-Indene, octahydro-, cis-	4551513	-126.25	-126.76
1-Methyl-bicyclo[3.1.0]-hexane	4625245	7.09	19.28
trans-syn-2-Methyldecalin	14398711	-209.23	-215.78
trans-Bicyclo[3.3.0]octane	5597897	-67.75	-67.76
cis-Bicyclo[5.3.0]decane	16189461	-132.45	-126.09
1,1’-Bicyclopropyl	5685461	128.01	128.93
1-Methylnorbornane	10052183	-92.74	-90.33
cis-Bicyclo[6.1.0]nonane	13757432	-31.22	-30.28
cis-Bicyclo[5.1.0]octane	16526902	-17.43	-17.42
1,4-Dimethyl-bicyclo[2.2.1]heptane	20454813	-127.27	-128.44
trans-2,3-Dimethyl-bicyclo[2.2.1]heptane	20558161	-106.58	-107.76

Continued on next page

Name	CAS	Observed (kJ mol^{-1})	SPARC
1,1'-Bicycloheptyl	23183111	-214.50	-216.94
Bicyclo[3.3.3]undecane	29415950	-94.25	-94.23
trans-(+)-Bicyclo-[6.1.0]nonane	39124793	-33.32	-34.25
Cyclopentylcycloheptane	42347488	-165.37	-172.69
Camphene	79925	-25.89	-21.34
Bicyclo[3.2.1]oct-2-ene	823029	15.69	14.48
2,5-Norbornadiene	121460	245.34	236.96
Azulene	275514	259.81	260.06
Azulene, 1,4-dimethyl-7-(1-methylethyl)-	489849	128.35	128.10
2-Methylenebicyclo[2.2.1]-heptane	497358	36.37	35.80
2-Norbornene	498668	89.66	92.37
Bicyclo[4.1.0]hept-2-ene, 3,7,7-trimethyl-	554610	28.24	25.04
Bicyclo(3.1.0)hex-2-ene	694019	156.83	155.96
Bicyclo[2.2.1]hept-2-ene, 2-methyl-	694928	44.45	49.72
Spiro[2,4]hepta-4,6-diene	765468	233.56	239.26
Bicyclo[2.2.2]oct-2-ene	931646	21.67	19.99
2-Methylene-bicyclo[2.2.2]-octane	2972205	-8.76	-3.35
Bicyclo[2.2.1]hept-2-ene, 5-ethenyl-	3048644	159.89	167.70
Bicyclo[2.2.0]hex-2-ene	3097630	261.40	257.66
Bicyclo[4.2.0]oct-7-ene, cis-	3806824	119.72	113.36
Bicyclo[3.2.1]octa-2,6-diene	4096951	157.86	159.07
2-Methyl-bicyclo[2.2.2]-oct-2-ene	4893134	-16.12	-14.63
Bicyclo(3.3.1)non-2-ene	6671665	-47.50	-47.49
Bicyclo[3.2.2]non-6-ene	7124869	-6.34	-6.31

Continued on next page

Name	**CAS**	**Observed (kJ mol^{-1})**	**SPARC**
Bicyclo[4.2.0]oct-1(6)-ene	10563118	95.40	97.53
3-Carene	13466789	21.25	25.04
Bicyclo[4,2,1]non-3-ene	16456330	0.42	0.43
Bicyclo[4.1.0]hept-3-ene	16554839	120.50	116.61
(Z)-5-Ethylidene-bicyclo[2.2.1]hept-2-ene	28304667	145.32	143.60
Bicyclo[2.2.0]hex-1(4)-ene	30830207	304.00	306.23
7-Methylenebicyclo[2.2.1]-heptane	31463351	59.91	65.35
7-Methylenebicyclo[4.1.0]hept-2-ene	36398966	250.92	260.42
cis-Bicyclo[4.3.0]nona-3,7-diene	38451182	109.59	102.49
Bicyclo[1.1.0]but-1(3)-ene	58208494	544.00	541.76
(E) 3,3'-Bis-(1-cyclohexenylidene)	132911343	92.00	92.05
(Z) 3,3'-Bis-(1-cyclohexenylidene)	132911354	93.20	92.05
Nortricyclene	279196	106.37	106.35
Adamantane	281232	-126.68	-129.37
Tricyclo[8.2.2.24,7]-hexadecane	283681	-161.66	-145.68
Benzvalene	659858	365.30	363.76
2-Methyladamantane	700561	-151.83	-158.46
1-Methyladamantane	768912	-175.86	-171.53
1,3,5,7-Tetramethyladamantane	1687361	-297.69	-298.02
trans-syn-trans-Perhydroanthracene	1755197	-243.21	-249.95
4,7-Methano-1H-indene, octahydro-	6004382	-65.68	-64.77
Tricyclo[5.2.1.02,6]decane	2825823	-73.70	-64.77
endo-Trimethylenenorbornane	2825834	-61.90	-56.82
Tricyclo[4.2.0.02,5]octane, anti-	13027753	206.39	213.45

Continued on next page

Name	CAS	Observed (kJ mol^{-1})	SPARC
Anti-tricyclo[4.1.0.0(2,4)]heptane	16782448	154.00	164.09
Perhydrotriquinacene	17760917	-107.31	-106.02
Protoadamantane	19026949	-85.90	-85.91
Tricyclo[3.2.1.01,5]octane	19074250	152.00	152.01
cis-syn-cis-Perhydroanthracene	19128780	-234.30	-213.21
Spiro[bicyclo[2.1.0]pentane...	19446685	286.45	277.14
2,2-Dimethyladamantane	19740342	-188.90	-183.59
trans-anti-trans-Perhydroanthracene	28071990	-220.66	-221.25
Tricyclo[4.2.0.02,5]octane, syn-	28636104	230.39	229.34
Dispiro[2.0.2.1]heptane	33475228	303.07	285.67
Tetra-tert-butyltetrahedrane	66809061	45.19	45.19
trans-1,2-dicyclopropylcyclopropane	150895640	181.57	186.96
cis-1,2-dicyclopropylcyclopropane	150895651	183.83	186.96
Bullvalene	1005512	303.63	303.64
endo-Tricyclo[3.2.1.02,4]oct-6-ene	3635947	239.00	234.23
exo-Tricyclo[3.2.1.02,4]oct-6-ene	3635958	208.13	214.37
Triquinacene	6053743	219.10	217.80
1,2-Dihydrotriquinacene	31678747	122.80	122.41
Tetrahydrotriquinacene	57595398	7.80	8.20
Cubane	277101	576.65	576.63
Quadricyclane	278068	353.61	353.60
Ethanediylidenecyclopenta[cd]pentalene	704029	84.77	84.77
Dicyclopropa[cd,gh]pentalene,octahydro-	765720	238.37	238.36
Tetracyclo[6.2.1.0(2,7).o(3,5)]undecane	1777442	57.64	57.23

Continued on next page

Name	**CAS**	**Observed (kJ mol^{-1})**	**SPARC**
Diadamantane	2292797	-145.90	-142.91
1-Methyldiadamantane	26460764	-166.70	-166.73
4-Methyldiadamantane	30545289	-182.10	-185.07
Trispiro[2.0.2.0.2.0]nonane	31561598	429.43	445.47
Tetracyclo[4.1.0.0(2,4).0(3,5)]heptane	50861262	370.00	371.54
Trispiro[2.0.0.2.1.1]nonane	50874243	434.09	422.95
Pentacyclo[6.3.1.13,6.02,7.09,11]tridecane	61140689	83.11	76.93
Pentacyclo[6.3.1.0.0.0]dodecane	82110701	188.93	179.23
painintheassane	86301942	406.65	399.92
10-Methylpentacyclotridecane	114056461	42.64	48.84
Tetraspiro[2.0.0.0.2.1.1.1]undecane	129872306	546.11	560.23
Benzene	71432	80.21	80.30
Benzene, hexamethyl-	87854	-91.14	-92.37
Benzene, 1,2-dimethyl-	95476	17.54	14.07
Benzene, 1,2,4-trimethyl-	95636	-14.56	-18.17
Benzene, 1,2,4,5-tetramethyl-	95932	-64.78	-52.16
Benzene, tert-butyl-	98066	-24.30	-24.68
Benzene, 1-(1,1-dimethylethyl)-4-methyl-	98511	-57.86	-56.92
Benzene, (1-methylethyl)-	98828	1.77	1.13
Benzene, 1,3-bis(1-methylethyl)-	99627	-77.85	-78.03
Benzene, 1-methyl-4-(1-methylethyl)-	99876	-29.94	-31.11
Benzene, 1,4-bis(1-methylethyl)-	100185	-76.98	-78.03
Ethylbenzene	100414	27.71	26.84
Benzene, propyl-	103651	6.25	5.74

Continued on next page

Name	CAS	Observed (kJ mol^{-1})	SPARC
Benzene, butyl-	104518	-13.46	-15.30
Benzene, decyl-	104723	-140.75	-139.87
Benzene, 1,4-diethyl-	105055	-23.69	-26.63
p-Xylene	106423	17.33	15.82
Benzene, 1,3-dimethyl-	108383	15.96	15.82
Benzene, 1,3,5-trimethyl-	108678	-17.23	-16.42
Toluene	108883	48.46	48.06
Benzene, 1,2-diethyl-	135013	-19.43	-17.79
Benzene, (1-methylpropyl)-	135988	-18.26	-18.88
Benzene, 1,3-diethyl-	141935	-24.57	-26.63
Benzene, 1,2,3,4-tetramethyl-	488233	-42.36	-43.49
Benzene, 1,2,3-trimethyl-	526738	-10.42	-14.71
Benzene, 1,2,3,5-tetramethyl-	527537	-42.99	-46.95
1-Methyl-2-iso-propylbenzene	527844	-25.19	-21.31
1-Methyl-3-iso-propylbenzene	535773	-30.72	-31.11
Benzene, (2-methylpropyl)-	538932	-22.44	-19.94
Benzene, 1-ethyl-2-methyl-	611143	-0.88	-1.88
Benzene, 1-ethyl-3-methyl-	620144	-3.62	-5.40
1-Ethyl-4-methylbenzene	622968	-4.26	-5.40
Benzene, pentamethyl-	700129	-73.43	-72.26
Benzene, 1-ethyl-2,4-dimethyl-	874419	-33.10	-34.12
Benzene, 1-ethyl-2,3-dimethyl-	933982	-31.35	-30.64
Benzene, 1-ethyl-3,5-dimethyl-	934747	-37.31	-37.64
Benzene, 4-ethyl-1,2-dimethyl-	934805	-34.63	-39.39

Continued on next page

Name	CAS	Observed (kJ mol^{-1})	SPARC
Benzene, 1,4-bis(1,1-dimethylethyl)-	1012722	-125.59	-129.67
Benzene, 1,3-bis(1,1-dimethylethyl)-	1014604	-126.58	-129.67
Benzene, 1-methyl-2-propyl-	1074175	-23.38	-20.86
Benzene, 1-methyl-3-propyl-	1074437	-25.79	-26.50
Benzene, 1-methyl-4-propyl-	1074551	-25.48	-26.50
Benzene, 1-(1,1-dimethylethyl)-3-methyl-	1075383	-58.38	-56.92
1,2,4-Tri-tert-butylbenzene	1459116	-174.42	-177.96
Benzene, 1,3,5-tri-tert-butyl-	1460022	-231.81	-234.65
Benzene, 2-ethyl-1,4-dimethyl-	1758889	-34.17	-34.12
9-Phenyleicosane	2398654	-349.61	-351.50
Benzene, 2-ethyl-1,3-dimethyl-	2870044	-31.61	-27.91
Heneicosane, 11-phenyl-	6703806	-374.16	-372.26
m-Xylene, 5-tert-butyl-	98191	-90.66	-89.16
Benzene, (1-methylethenyl)-	98839	116.93	113.69
Styrene	100425	145.90	147.83
(Z)-1-Phenylpropene	766905	121.40	120.09
Benzene, 1-propenyl-, (E)-	873665	117.21	113.99
trans-1-Phenyl-3,3-dimethyl-but-1-ene	3846660	34.88	37.95
Benzyne	462806	440.00	407.96
Phenylacetylene	536743	306.60	317.89
Benzene, 1-butynyl-	622764	248.60	251.68
Benzene, 1-propynyl-	673325	268.20	275.55
Naphthalene, 1-methyl-	90120	117.88	113.28
Naphthalene	91203	135.04	140.41

Continued on next page

Name	CAS	Observed (kJ mol^{-1})	SPARC
Naphthalene, 2-methyl-	91576	107.67	108.17
Biphenyl	92524	161.34	167.59
Diphenylmethane	101815	164.14	158.80
Bibenzyl	103297	124.67	138.53
Tetralin	119642	27.87	26.15
Indane	496117	62.52	54.41
Naphthalene, 1,8-dimethyl-	569415	95.46	104.07
Naphthalene, 2,3-dimethyl-	581408	64.70	74.18
Naphthalene, 2,6-dimethyl-	581420	65.42	75.93
Naphthalene, 2,7-dimethyl-	582161	64.95	75.93
1,1-Diphenylethane	612000	133.98	133.83
1,1'-Biphenyl, 3,3'-dimethyl-	612759	92.22	103.11
4,4'-Dimethylbiphenyl	613332	94.15	103.11
4-Methyldiphenylmethane	620837	133.77	126.56
1,1'-Biphenyl, 2-methyl-	643583	152.72	144.50
1,1'-Biphenyl, 4-methyl-	644086	127.65	135.35
Bicyclo[4.2.0]octa-1,3,5-triene	694871	203.20	223.16
Benzene, cyclohexyl-	827521	-16.40	-19.24
1-Isopropyl-6-methylindane	828977	-45.56	-50.72
Benzene, cyclopropyl-	873494	154.07	155.97
1,1,4,6-Tetramethylindane	941606	-63.48	-67.99
Naphthalene, 1,2,3,4-tetrahydro-1-methyl-	1559815	-0.47	-1.17
1H-Indene, 2,3-dihydro-4,6-dimethyl-	1685821	-1.42	-10.75
1,4,5,8-Tetramethylnaphthalene	2717397	64.11	67.74

Continued on next page

Name	**CAS**	**Observed (kJ mol^{-1})**	**SPARC**
Naphthalene, 2-(1,1-dimethylethyl)-	2876359	44.03	35.42
1,1-Dimethyl-6-tert-butylindan	3605310	-102.39	-107.82
1H-Indene, 2,3-dihydro-1,1-dimethyl-	4912929	3.35	-2.83
1,1,4,6,7-Pentamethylindan	6682673	-101.12	-90.72
1H-Indene, 2,3-dihydro-4,7-dimethyl-	6682719	-3.31	-11.43
2,5-Dimethyldiphenylmethane	13540506	101.60	104.94
Naphthalene, 1-ethyl-1,2,3,4-tetrahydro-	13556586	-22.42	-21.42
1-Cyclopropyl-2,4,6-trimethylbenzene	26269590	69.90	69.36
1-Cyclopropyl-2-methylbenzene	27546469	129.01	127.39
1-Cyclopropyl-2,4-dimethylbenzene	27546470	95.62	95.15
1-Cyclopropyl-4-isopropylbenzene	27546492	74.36	76.81
2,5,8-Trimethyltetralin	30316177	-65.39	-66.34
1-Ethyl-8-methylnaphthalene	61886713	96.58	89.98
1-Isopropyl-8-methylnaphthalene	81603443	75.66	71.93
Indene	95136	159.94	176.84
(E)-Stilbene	103300	221.85	244.52
(Z)-Stilbene	645498	259.02	251.31
2,5-Diphenyl-1,5-hexadiene	7283490	285.00	273.63
2,6-Diphenyl-1,6-heptadiene	27905653	259.00	252.13
1-(Methylene)-2-phenylcyclopropane	29817092	292.00	299.69
Diphenylethyne	501655	385.00	399.90
Acenaphthene	83329	145.53	141.18
Phenanthrene	85018	201.35	197.88
9H-Fluorene	86737	163.75	159.83

Continued on next page

Name	CAS	Observed (kJ mol^{-1})	SPARC
p-Terphenyl	92944	256.99	255.87
o-Terphenyl	84151	273.77	261.86
m-Terphenyl	92068	257.18	257.46
Anthracene	120127	219.62	215.80
Biphenylene	259790	405.53	395.55
1H-Cyclopropa[b]naphthalene	286851	433.12	433.12
Triphenylmethane	519733	267.30	261.75
Phenanthrene, 9,10-dimethyl-	604831	145.50	145.49
Anthracene, 9,10-dihydro-	613310	154.91	152.17
Phenanthrene, 9,10-dihydro-	776352	140.07	142.75
Phenanthrene, 4-methyl-	832644	191.51	188.10
Phenanthrene, 1,2,3,4-tetrahydro-	1013087	92.30	91.12
Anthracene, 1,2,3,4,5,6,7,8-octahydro-	1079716	-33.94	-28.01
Phenanthrene, 2,7-dimethyl-	1576698	132.74	133.40
Di-1,4-xylylene	1633223	247.65	236.81
Di-1,3-xylylene	2319973	169.58	174.70
9H-Fluorene, 9-methyl-	2523377	141.59	133.29
Paracyclophane -[3.3]	2913248	123.61	123.69
Phenanthrene, 4,5-dimethyl-	3674699	184.45	178.32
6,6-Paracyclophane	4384230	-64.75	-63.66
2,2-Metaparacyclophane	5385364	215.99	205.75
1,8-Paracyclophane	6169944	29.00	27.25
Acenaphthylene	208968	256.47	252.67
Benz[a]anthracene	56553	282.43	277.75

Continued on next page

Name	CAS	Observed (kJ mol^{-1})	SPARC
Benz a anthracene, 7,12-dimethyl-	57976	268.24	272.23
Naphthacene	92240	281.33	291.19
Pyrene	129000	209.32	205.86
Benzo[c]phenanthrene	195197	297.96	297.31
Perylene	198550	307.26	297.38
Triphenylene	217594	266.34	271.03
Chrysene	218019	262.74	264.31
Benz[a]anthracene, 1,12-dimethyl-	313746	258.51	265.21
Triptycene	477758	337.94	330.77
Naphthalene, 1-(2-naphthalenylmethyl)-	611483	285.08	283.70
1,1':3',1"-Terphenyl, 5'-phenyl-	612715	349.09	348.52
Tetraphenylmethane	630762	379.56	387.74
sym-Tetraphenylethane	632508	349.37	350.29
Pyrene, 4,5,9,10-tetrahydro-	781179	117.31	116.80
Naphthacene, 5,12-dihydro-	959024	217.80	212.28
9,9'-Bianthracene	1055238	496.82	501.17
Anthracene, 9,10-diphenyl-	1499101	468.49	459.25
9,9'-Bi-9H-fluorene	1530127	343.29	356.24
Tritetralin	1610395	-76.03	-78.53
Benzyltriphenylmethane	2294942	366.85	367.77
Benzo[c]phenanthrene, 1,12-dimethyl-	4076431	275.72	277.75
9,9'-Dimethyl-9,9'-bifluorenyl	15300820	348.26	349.37
Pentacyclohexacosa-nonane	35117216	388.49	388.77
namewontfitane	56818065	164.82	177.19

Continued on next page

Name	CAS	Observed (kJ mol^{-1})	SPARC
Coronene	191071	291.49	295.93

B.2 Halogenated Hydrocarbons

Table B.2: Observed and calculated values of halogenated hydrocarbons

Name	CAS	Observed (kJ mol^{-1})	SPARC
Methyl fluoride	593533	-234.30	-248.04
Methyl chloride	74873	-85.90	-91.61
Methane, difluoro-	75105	-450.66	-458.00
Ethyl chloride	75003	-109.00	-113.80
Ethane, 1,1-difluoro-	75376	-497.00	-502.51
Methane, trifluoro-	75467	-695.40	-683.00
Propane, 2-chloro-	75296	-144.00	-143.96
Propane, 1-chloro-	540545	-131.20	-134.48
Ethane, 1-chloro-1-fluoro-	1615754	-313.40	-313.39
Ethane, 1,1,1-trifluoro-	420462	-748.70	-745.06
1,1,2-Trifluoroethane	430660	-691.00	-682.72
Methylene chloride	75092	-96.84	-97.19

Continued on next page

Name	CAS	Observed (kJ mol^{-1})	SPARC
Methane, chlorodifluoro-	75456	-483.00	-473.30
Carbon tetrafluoride	75730	-934.30	-930.29
Butane, 2-chloro-	78864	-166.66	-163.87
Butane, 1-chloro-	109693	-156.51	-155.21
Propane, 2-chloro-2-methyl-	507200	-183.73	-181.28
Methyl bromide	74839	-38.48	-37.73
Ethane, 1,1-dichloro-	75343	-127.60	-130.56
Ethane, 1,2-dichloro-	107062	-136.25	-134.51
Methane, chlorotrifluoro-	75729	-707.10	-698.73
Butane, 1-chloro-3-methyl-	107846	-179.50	-181.60
Pentane, 1-chloro-	543599	-177.04	-175.96
Ethane, bromo-	74964	-68.45	-64.11
Propane, 1,2-dichloro-	78875	-162.80	-164.10
Propane, 2,2-dichloro-	594207	-174.06	-171.38
Chloroform	67663	-103.53	-98.50
Dichlorodifluoromethane	75718	-477.00	-482.04
Propane, 2-bromo-	75263	-99.00	-97.96
Propane, 1-bromo-	106945	-92.78	-84.81
1,4-Dichlorobutane	110565	-181.02	-182.52
Butane, 1,3-dichloro-	1190223	-194.87	-190.83
Methane, bromodifluoro-	1511622	-425.30	-421.05
Propane, 3-chloro-1,1,1-trifluoro-	460355	-779.70	-793.08
Ethane, 1,1,1-trichloro-	71556	-144.60	-129.77
Ethane, 1,1,2-trichloro-	79005	-148.76	-145.08

Continued on next page

Name	CAS	Observed (kJ mol^{-1})	SPARC
Butane, 2-bromo-	78762	-121.97	-117.71
Butane, 1-bromo-	109659	-107.62	-105.56
Propane, 2-bromo-2-methyl-	507197	-132.40	-138.42
Trichloromonofluoromethane	75694	-268.30	-280.21
Ethane, hexafluoro-	76164	-1344.00	-1351.42
Methyl iodide	74884	13.31	15.56
1-Fluorononane	463183	-425.87	-427.88
Propane, 1,2,3-trichloro-	96184	-182.20	-184.81
Octane, 1-chloro-	111853	-240.94	-238.25
Methane, bromotrifluoro-	75638	-649.40	-652.26
Pentane, 1-bromo-	110532	-129.57	-126.32
Carbon tetrachloride	56235	-97.37	-93.25
Iodoethane	75036	-9.42	-7.79
Hexane, 1-bromo-	111251	-148.96	-147.08
3,3-Dichloro-1,1,1-trifluoropropane	460695	-807.19	-809.57
Ethane, 1,1,2,2-tetrachloro-	79345	-149.65	-151.78
Ethane, 1,1,1,2-tetrachloro-	630206	-154.28	-150.92
1-Chloro-1,1,3,3,3-pentafluoropropane	460924	-1158.91	-1158.92
Propane, 2-iodo-	75309	-42.31	-38.05
Propane, 1-iodo-	107084	-30.77	-28.52
1,2-Dichloro-1,1,2,2-tetrafluoroethane	76142	-940.56	-939.51
Heptane, 1-bromo-	629049	-168.33	-167.84
Propane, 1,1,1,3-tetrachloro-	1070786	-162.55	-175.36
Propane, 1,2,2,3-tetrachloro-	13116535	-199.96	-196.29

Continued on next page

Name	CAS	Observed (kJ mol^{-1})	SPARC
Tert-butyl iodide	558178	-72.00	-74.02
Ethane, 1,1,2-trichloro-1,2,2-trifluoro-	76131	-729.09	-722.87
Ethane, 1,1-dibromo-	557915	-28.51	-28.50
Propane, octafluoro-	76197	-1784.70	-1769.78
Dodecane, 1-fluoro-	334689	-489.51	-490.18
Ethane, 1-bromo-1,1-difluoro-	420439	-474.30	-474.29
Ethane, 1,1-difluoro-1-iodo-	420473	-410.00	-409.95
Octane, 1-bromo-	111831	-190.33	-188.61
Methane, trifluoroiodo-	2314978	-587.80	-587.83
Halothane	151677	-693.27	-693.23
1-Bromo-2-chloro-1,1,2-trifluoroethane	354063	-649.30	-649.30
Methane, bromotrichloro-	75627	-42.00	-42.01
1,1,1-Trichloro-3,3,3-trifluoropropane	7125840	-799.44	-801.48
Ethane, pentachloro-	76017	-144.35	-151.98
Dodecane, 1-chloro-	112527	-321.51	-321.31
1,1,1-Trifluoro-2-iodoethane	353833	-648.70	-648.68
Butane, 1,3-dibromo-	107802	-97.44	-95.11
Butane, 1,4-dibromo-	110521	-83.95	-83.23
Butane, 1,2-dibromo-	533982	-93.11	-97.28
Propane, 1,2-dibromo-2-methyl-	594343	-105.34	-113.88
1,3-Dibromoisobutane	28148041	-92.15	-88.61
Tetradecane, 1-fluoro-	593339	-534.50	-531.71
1,2-Dichlorohexafluoropropane	661972	-1350.16	-1356.08
Pentane, 1,2-dibromo-	3234499	-112.86	-117.92

Continued on next page

Name	CAS	Observed (kJ mol^{-1})	SPARC
Ethane, hexachloro-	67721	-143.50	-140.97
Dodecane, 1-bromo-	143157	-269.53	-271.67
Methane, tribromo-	75252	52.71	51.34
Methane, diiodo-	75116	116.72	109.33
Ethane, 1,2-diiodo-	624737	62.70	62.23
Octadecane, 1-chloro-	3386332	-450.11	-445.90
1,2-Diiodopropane	598298	35.60	32.83
Propane, 1,3-diiodo-	627316	48.47	50.09
Hexadecane, 1-bromo-	112823	-353.47	-354.73
Butane, 1,4-diiodo-	628217	32.94	29.32
1,2-Diiodobutane	53161721	12.30	12.78
Carbon tetrabromide	558134	80.75	81.42
Heptane, hexadecafluoro-	335579	-3383.70	-3389.07
Methane, triiodo-	75478	253.63	263.47
Methane, tetraiodo-	507255	469.10	465.41
Perfluoro-2,7-dimethyloctane	3021634	-4636.95	-4635.44
Ethene, fluoro-	75025	-136.00	-133.96
Ethene, chloro-	75014	29.00	23.04
Ethene, 1,1-difluoro-	75387	-334.00	-333.92
Ethene, 1,2-difluoro-, (E)-	1630780	-290.40	-288.39
Ethene, trifluoro-	359115	-490.40	-491.95
Propene, 3,3,3-trifluoro-	677214	-614.20	-614.29
Ethene, 1,1-dichloro-	75354	-2.54	6.65
Ethene, 1,2-dichloro-, (Z)-	156592	4.18	5.36

Continued on next page

Name	CAS	Observed (kJ mol^{-1})	SPARC
Ethene, 1,2-dichloro-, (E)-	156605	5.02	-6.39
Ethene, tetrafluoro-	116143	-661.00	-663.52
Ethene, bromo-	593602	79.20	79.19
2,3-Dichloropropene	78886	-36.13	-31.75
Ethene, chlorotrifluoro-	79389	-523.00	-516.43
Trichloroethylene	79016	-10.53	-11.03
1-Propene, 1,2,3-trichloro-	96195	-57.39	-49.33
Tetrachloroethylene	127184	-11.30	-15.68
(Z)-1,2-Diiodoethylene	590261	207.40	210.26
(E)-1,2-Diiodoethylene	590272	207.40	204.54
Perfluoro-2-methyl-2-pentene	1584038	-2490.64	-2501.90
Cyclohexane, fluoro-	372463	-339.92	-339.98
1,1,2,2-Tetrafluorocyclopropane	3899716	-650.00	-643.01
Cyclohexane, chloro-	542187	-167.95	-164.38
Cyclopropane, hexafluoro-	931919	-978.20	-982.86
Octafluorocyclobutane	115253	-1488.00	-1487.86
Cyclohexane, 1,2-dibromo-	5401627	-103.28	-103.31
Lindane	58899	-297.32	-298.71
α-Lindane	319846	-311.52	-304.54
β-Lindane	319857	-310.99	-316.19
δ-Lindane	319868	-309.38	-310.36
Cyclohexane, dodecafluoro-	355680	-2368.90	-2366.76
Perfluoromethylcyclohexane	355022	-2895.99	-2905.79
Perfluoroethylcyclohexane	335217	-3301.16	-3294.11

Continued on next page

Name	CAS	Observed ($kJ mol^{-1}$)	SPARC
Decafluorocyclohexene	355759	-1906.60	-1906.60
trans-Perfluorobicyclo[4,3.0]nonane	75240061	-3210.42	-3209.89
cis-Perfluorobicyclo[4.3.0]nonane	75262872	-3212.20	-3213.86
cis-Perfluorodecalin	60433116	-3619.39	-3619.42
trans-Perfluorodecalin	60433127	-3644.50	-3643.45
Perfluorobicyclo-[4.4.0]-dec-1,6-ene	54939047	-3205.42	-3205.46
Benzene, fluoro-	462066	-117.83	-116.35
Benzene, (fluoromethyl)-	350505	-127.01	-134.87
Benzene, 1-fluoro-4-methyl-	352329	-148.53	-148.59
Benzene, chloro-	108907	50.48	52.23
Benzene, 1,2-difluoro-	367113	-286.17	-287.10
Benzene, 1,3-difluoro-	372189	-312.68	-313.00
Benzene, 1,4-difluoro-	540363	-310.71	-313.00
Benzene, (chloromethyl)-	100447	18.25	21.54
Benzene, 1-chloro-2-ethyl-	89963	0.44	0.53
Benzene, 1-chloro-3-ethyl-	620166	-1.50	-1.16
Benzene, 1-chloro-4-ethyl-	622980	-0.55	-1.16
Benzene, (1-chloroethyl)-	672651	-5.76	-6.76
Benzene, (trifluoromethyl)-	98088	-599.10	-598.59
Benzene, 1,2-dichloro-	95501	32.76	23.28
Benzene, 1,4-dichloro-	106467	24.60	24.15
Benzene, 1,3-dichloro-	541731	25.89	24.15
Benzene, 1,2,4,5-tetrafluoro-	327548	-656.67	-654.50
1,1-Difluoro-3-phenylpropane	1.46E+08	-418.98	-409.48

Continued on next page

Name	CAS	Observed (kJ mol^{-1})	SPARC
Benzene, bromo-	108861	103.80	109.97
1,1,1-Trifluoro-2-phenylethane	21249934	-629.67	-627.72
1-Fluoro-3-(trifluoro-methyl)benzene	401809	-794.59	-795.08
Benzene, pentafluoro-	363724	-806.00	-809.42
m-Tert-butyl chlorobenzene	3972552	-54.71	-52.68
1-Chloro-4-(1,1-dimethylethyl)benzene	3972563	-53.21	-52.68
Benzene, 1-tert-butyl-2-chloro-	7073985	-33.12	-33.99
Benzene, (bromomethyl)-	100390	71.26	75.29
Benzene, 1,2,3-trichloro-	87616	8.20	2.06
Benzene, 1,3,5-trichloro-	108703	-2.60	-3.92
Benzene, 1,2,4-trichloro-	120821	-8.05	-4.79
Benzene, pentafluoromethyl-	771562	-852.28	-852.28
Benzene, hexafluoro-	392563	-965.78	-964.35
1-Bromo-4-chlorobenzene	106398	82.58	81.98
Benzene, chloropentafluoro-	344070	-818.24	-817.91
Benzene, iodo-	591504	169.05	166.94
Benzene, 1,2,4,5-tetrachloro-	95943	-32.62	-33.73
Benzene, 1,2,3,4-tetrachloro-	634662	-25.40	-19.17
Benzene, 1,2,3,5-tetrachloro-	634902	-34.90	-26.02
Benzene, 1-iodo-2-methyl-	615372	133.14	135.43
Benzene, (iodomethyl)-	620053	127.30	128.41
Benzene, 1-iodo-4-methyl-	624317	126.49	134.70
Benzene, 1-iodo-3-methyl-	625956	136.55	134.70
2,4-Di-t-butyl chlorobenzene	80438659	-137.14	-138.98

Continued on next page

Name	**CAS**	**Observed (kJ mol^{-1})**	**SPARC**
3,5-Di-tert-butyl chlorobenzene	80438671	-160.43	-157.66
Benzene, pentafluoro(trifluoromethyl)-	434640	-1285.92	-1285.91
1,2,4,5-Tetrachloro-3,6-dimethylbenzene	877101	-93.18	-93.17
Benzene, pentachloro-	608935	-40.00	-40.39
Benzene, hexachloro-	118741	-44.70	-47.05
Benzene, pentafluoroiodo-	827156	-549.00	-549.01
1,2-Diiodobenzene	615429	256.77	256.60
Benzene, 1,4-dichloro-2-ethenyl-	1123848	92.14	95.24
Fluorodiphenylmethane	579555	-39.11	-29.68
2,2'-Difluorobiphenyl	388829	-229.65	-229.65
1,1'-Biphenyl, 4,4'-difluoro-	398232	-227.56	-225.39
Naphthalene, 1-bromo-	90119	182.30	174.95
2-Bromonaphthalene	580132	168.31	170.08
1,1-Difuoro-1,2-diphenylethane	350629	-260.50	-278.46
1,1'-Biphenyl, 4,4'-dichloro-	2050682	120.00	111.60
1,1'-Biphenyl, 2,2'-dichloro-	13029088	127.90	129.19
1,1,1-Trifluoro-2,2-diphemylethane	384941	-515.68	-503.44
1,1,2-Trifluoro-1,2-diphenylethane	68936776	-462.20	-458.27
1-Iodonaphthalene	90142	239.77	231.92
Naphthalene, 2-iodo-	612555	225.37	227.05
1,2-Diphenyltetrafluoroethane	425321	-689.00	-675.08
Perfluorobiphenyl	434902	-1264.20	-1264.21

Appendix C

The heat of formation PROLOG code

C.1 HF.pro

```
/**********************************************************************
The call for when running in batch mode
**********************************************************************/
heat :-
batch_mode,
hf_clean(NewList),
hf_aux(NewList,0,HF),
assert(value_calculated(HF)).

/***********************************************************************
This handles smiles strings with '/' or '\' in them (cis-trans)
***********************************************************************/
hf(HF) :-
write('Give me a SMILES:'),nl,
cis_trans_read(user_input,Smiles),
hf_top(Smiles,HF).

/**********************************************************************
The call for running smiles strings that are not cis-trans
***********************************************************************/
hf(Smiles,HF) :-
write('ATTENTION!!'),nl,
write('ATTENTION!!'),nl,
write('THIS WILL NOT WORK FOR CIS-TRANS, USE OTHER HF'),nl,
hf_top(Smiles,HF).

/********************************************************************
finds the proper data     (_top2)
gets the hf           (_aux)
prints out the parts to each calculation  (_values & _values2)
*********************************************************************/
hf_top(Smiles,HF) :-
```

```
execute(prepare_for_property(nul,Smiles),ResTop),
        ResTop = true,
        !,
hf_clean(NewList),
hf_aux(NewList,0,HF),
nl,
        findall([Name,Reac,Value],data_tad(Name,Reac,Value),Parts),
        hf_values(Parts),
        hf_values2(Parts).
hf_top(_,HF) :-
HF = 'Error!!!'.

/*********************************************************************
removes all the asserted facts
gets all of the reactophore facts
fixes the db for aromatics
gets the hf
*********************************************************************/
hf_clean(NewList):-
retractalls(data_tad(_,_,_)),      %remove valueDB
retractalls(hf_bridge_atom(_)),    %remove hf_bridge atomsDB
retractalls(chiral_atom_info(_,_,_,_,_,_)),
%remove chiral_atom_info (replace all other CA data)

retractalls(chiral_pairs(_)),      %remove chiral_pairsDB
retractalls(chiral_sense(_,_,_)),  %remove chiral_senseDB

%% retractalls(chiral_subsense(_,_,_)),  %remove chiral_subsenseDB
%% retractalls(chiral_size(_,_,_)),   %remove chiral_sizeDB

retractalls(done_chiral_pairs(_)), %remove done_chiralsDB
retractalls(hf_structure(_,_,_)),  %remove structureDB
findall(reactophore_fact(A,B,C,D,E), reactophore_fact(A,B,C,D,E), List),
alter_hf_db(List,NewList).

/********************************************************************
        Change database so that if you have an aromatic cmpd, it calls this
routine to change the cmpd so it is described as an aromatic, not
a large number of ethylenics.

Keyword:  annulene
*********************************************************************/
alter_hf_db(Reacs,NewReac):-
findall(conj_ring(K,RA),conj_ring(K,RA),Z),
alter_hf_db_aux(Z,Reacs,NewReac).
alter_hf_db_aux([],Reacs,Reacs).
alter_hf_db_aux([conj_ring(K,RA)|T],_Reacs,NewReac):-
        length(RA,Len),
        N is (Len - 2) / 4,
        IntN is integer(N),
        X is IntN * 4 + 2,
X =:= Len,
findall(reactophore_fact(A,e,[C1,C2],D,E),
(reactophore_fact(A,e,[C1,C2],D,E),
         intersects([C1,C2],RA,[_,_])),DBList),
findall(reactophore_fact(L,M,NN,O,P),(reactophore_fact(L,M,NN,O,P),
      \+ memberchk(reactophore_fact(L,M,NN,O,P),DBList)
     ),StrippedReac),
retract(ring(K,_)),
retract(conj_ring(K,_)),
!,
build_long_arom(DBList,LongArom), %reactophore_fact
alter_hf_db_aux(T,[LongArom|StrippedReac],NewReac).
alter_hf_db_aux([_|T],Reacs,NewReac):-
alter_hf_db_aux(T,Reacs,NewReac).

build_long_arom(DBList,LongArom):-
sparc_gensym(arom,Type),
length(DBList,Len),
X is Len * 2,
name(X,Y),
name(Name,[108|Y]),
hf_build_atom_list(DBList,[],Atoms),
atom_types(Atoms,AtomTypes),
hf_build_atom_subs(Atoms,Atoms,[],SubsList),
LongArom =.. [reactophore_fact,Type,Name,Atoms,AtomTypes,SubsList],
assert(LongArom),
assert(bridge_atoms(Type,[])),
assert(buried_atoms(Type,[])),
findall(W,(bonded_pair(W,E,R,T),
   memberchk(reactophore_fact(W,_,_,_,_),DBList),
   memberchk(reactophore_fact(T,_,_,_,_),DBList),
   retract(bonded_pair(W,E,R,T))),_),
hf_repair_bonded_pair(DBList,Type),
findall(XX,(member(XX,DBList),
    retract(XX)),_).

hf_repair_bonded_pair(DBList,Type):-
bonded_pair(W,E,R,T),
reactophore_fact(W,e,A,S,D),
memberchk(reactophore_fact(W,e,A,S,D),DBList),
prove_once(assert(bonded_pair(Type,E,R,T))),
retract(bonded_pair(W,E,R,T)),
fail.
hf_repair_bonded_pair(DBList,Type):-
bonded_pair(W,E,R,T),
reactophore_fact(T,e,A,S,D),
memberchk(reactophore_fact(T,e,A,S,D),DBList),
prove_once(assert(bonded_pair(W,E,R,Type))),
retract(bonded_pair(W,E,R,T)),
fail.
hf_repair_bonded_pair(_,_).

hf_build_atom_list([],A,A).
hf_build_atom_list([reactophore_fact(_,_,[C1,C2],_,_)|T],In,Atoms):-
hf_build_atom_list(T,[C1,C2|In],Atoms).

/***************************************************************************
        Writes out the result from each part of the calculation.
                _values prints each part on the screen
                _values2 makes sure that we got all of the values
**************************************************************************/
hf_values([]):-
        write('Done.'),nl.
hf_values([[Name,'_',Value]|T]):-
        write('The whole molecule has a '), write(Name), write(' of '),
write(Value), write(' kJ.'),nl,
        hf_values(T).
hf_values([[Name,Reac,Value]|T]):-
        reac_type(Reac,_,Type),
        write('The reactophore '),write(Reac),write('('),
        write(Type),write(')'),
        write(' has a '),write(Name),write(' of '),write(Value),
        write(' kJ.'),nl,
        hf_values(T).

hf_values2(Parts):-
        findall(X,(member(Y,Parts),
                   Y = [_,_,X]),List),
        sumlist(List,Hf),
        write('This should match what is below: '),write(Hf),nl.

/*************************************************************************
        The starting place where all the real work gets done.
```

```
*******************************************************************/
hf_aux([],Part,HF) :-
        findall(X,ring(_,X),Rings),
        gensort(Rings,longer,Sorted),
        superset_filter(Sorted,Filteredx),
structure_separator(Filteredx,[],[],Groups),
hf_filters(Groups,[],Filtered),

flatten(Filtered,Flat),
delete_duplicates(Flat,RingAtoms),

hf_ring_correct(Filtered,Filtered,RingAtoms,0,Correct),      %for rings
hf_data(steric,'_',Correct),

hf_homoarom(Filtered,RingAtoms,HA),    %for molecules w/ homoaromaticity
hf_data(resonance,'_',HA),

% hf_ethy_ring_correct(Filtered,RingAtoms,0,ERC),%molecules w/ endo ethys
% NERC is ERC * -1,
% hf_data(stericerc,'_',NERC),

hf_near_methyl(RingAtoms,Near),%for methyl groups w/ 3subs connected to same
hf_data(steric,'_',Near),

 %for aroms or rings joined by 2 chains of atoms
hf_ring_chain(Filteredx,Filtered,RC),
hf_data(steric,'_',RC),

hf_resonance3(Res3),%for arom-ethy connected together--!!!CHECK DATA
hf_data(resonance,'_',Res3),

hf_chiral(Filtered,Chiral),
hf_data(steric,'_',Chiral),

%% HF is Part + Correct + HA + Near + RC + Chiral.
HF is Part + Correct + HA + Near + RC + Chiral + Res3.
%res3 is temporary(I hope)

hf_aux([Reac|Rest],Part,HF) :-
reac_hf(Reac,PartHF),
NewPart is Part + PartHF,
hf_aux(Rest,NewPart,HF).

/******************************************************************
For molecules that have ethylenics endo in them.  This is a steric
```

```
correction b/c as more ethys get added to a ring, and are next to
another ethy, the ring strain goes up more than can be accounted for
by a linear correction. It increases as a parabola w/ 6-membered ring
at the minimum.

!! If there are any e's exo in the ring next to an endo, then don't get
a correction, unless the exo is bound to only 1 of the endos (I think)

*********************************************************************/
hf_ethy_ring_correct([],_,Out,Out).
hf_ethy_ring_correct([H|T],_,Part,ERC):-
findall(NDO,(reactophore_fact(NDO,e,[A,B],_,_),%List of E's endo in ring
     intersects([A,B],H,[_,_])),ListNDO),
ListNDO = [_,_|_],
findall(XXX-YYY,(member(XXX,ListNDO),
         member(YYY,ListNDO),
         XXX \= YYY,
         pair(XXX,YYY)),Pairs),
Pairs \= [],
findall(XO,(reactophore_fact(XO,e,[A,B],_,_),% List of E's exo in ring
    intersects([A,B],H,[_]),% that are paired to 2 endos
    member(NDO,ListNDO),
    member(NDO2,ListNDO),
    NDO2 \= NDO,
    pair(XO,NDO),
    pair(XO,NDO2)),ListExo),
length(ListExo,LenXO),
LenXO < 1,
length(ListNDO,NumNDO),
Num is NumNDO-1,
length(H,Len),

/**
hf_erc(a,A),
hf_erc(b,B),
hf_erc(c,C),
Basic is (A*(Len*Len))+(B*Len) + C,
***/

hf_erc(Len,Basic),
NewPart is Part + Basic * Num,
hf_ethy_ring_correct(T,_,NewPart,ERC).
hf_ethy_ring_correct([_|T],_,Part,ERC):-
hf_ethy_ring_correct(T,_,Part,ERC).
```

```
/**************************************************************************
For molecules w/ homoaromaticity.  Like triquinacene: val = 224.
Since this is the only one going to set parameter by hand and not
train until get more in set.

See JACS 1986 Liebman,Paquette,Peterson,Rogers
**************************************************************************/
hf_homoarom(Filtered,RingAtoms,HA):-
findall(Atom,(member(Atom,RingAtoms),
      atom_specs(_,Atom,_,_,List),
      length(List,LL),
              LL > 2,
      findall(X,(member(X,Filtered),
             length(X,L),
     L > 3,
     intersects(List,X,[_,_])),[A,B,C]),
      intersects(A,B,[_,_]),
      intersects(A,C,[_,_]),
      intersects(B,C,[_,_])),Flats),
Flats \= [],
hf_homoarom_aux(Flats,Filtered,RingAtoms,0,HA).
hf_homoarom(_,_,0).

hf_homoarom_aux([],_,_,In,In).
hf_homoarom_aux([H|T],Filtered,RingAtoms,Part,HA):-
findall(X,(reactophore_fact(X,e,[A,B],_,_),
   member(Ring,Filtered),
   memberchk(A,Ring),
   memberchk(B,Ring),
   memberchk(H,Ring)),[_,_,_]),
hf_homarom(pp,Val),
NewPart is Part + Val,
hf_homoarom_aux(T,Filtered,RingAtoms,NewPart,HA).
hf_homoarom_aux([_|T],Filtered,RingAtoms,Part,HA):-
hf_homoarom_aux(T,Filtered,RingAtoms,Part,HA).


/*******************************************************************
resonance3:  Correction for molecules w/ aromatic and ethylenic

Value is the Num*Value

!!!! This may be justified only by bad data, but since I don't have
any good data to back it up, I'm going w/ this
```

```
Check the Roth-'91 paper for errors!
*******************************************************************/
hf_resonance3(Value):-
findall(A-B, (reactophore_fact(A,_,_,_,_),
                reac_type(A,_,Pi),
        memberchk(Pi,[aromatic]),
      pair(A,B),
%%               reac_type(B,_,Pi2),
      n_o_s_type(B,ethylenic)),ListPair),
length(ListPair,Pairs),
resonance3(pp,Val),
Value is Pairs*Val.

/*******************************************************************
Correction for molecules joined by chains of atoms:
_chain1 is for aromatics
_chain2 is for nonaromatics ie cyclohexane
*******************************************************************/
hf_ring_chain(Filteredx,Filtered,RC):-
hf_ring_chain1(Filteredx,Filtered,AromRC),
hf_ring_chain2(Filteredx,Filtered,RingRC),
RC is AromRC + RingRC.

/*******************************************************************
Correction for molecules w/ two aroms joined by chains of
atoms:
only deals w/ molecules that are joined together like a ring
the [] checks that Filtered is []
the [H|T] are the rings found from above...but we
     aren't using them, just making sure that rings are
     present

rc_position deals w/ the position off of each arom:
position_aux: finds the number in each location off each arom
multiplies each location by a value to get correction
rc_aux2 deals w/ the chain length of atoms between the aroms:
really adds up the length of the compound (like a big ring
correction) and gets a value based on that
**********************************************************************/
hf_ring_chain1([],_,0).
hf_ring_chain1(_,[_|_],0).
hf_ring_chain1([_|_],_,Correct) :-
findall(A,(reactophore_fact(A,_B,_C,_D,_E),
   reac_type(A,_,aromatic)),Aroms),
hf_rc_position(Aroms,_Ortho,_Meta,_Para,PositCorrect),
hf_rc_chain(Aroms,Aroms,[],ChainLenCorrect),
Correct is PositCorrect + ChainLenCorrect.

hf_rc_position(Aroms,Ortho,Meta,Para,PC):-
hf_rc_position_aux(Aroms,Aroms,0,Ortho,0,Meta,0,Para,0,OT),
hf_rc(ortho,ValO),
hf_rc(meta,ValM),
hf_rc(para,ValP),
PC is Ortho*ValO + Meta*ValM + Para*ValP + OT.

/*******************************************************************
rc_position_aux finds the number in each position off of each arom:
*******************************************************************/
hf_rc_position_aux([],_,O,O,M,M,P,P,OT,OT).
hf_rc_position_aux([H|T],Aroms,IO,O,IM,M,IP,P,PartOT,OT):-
findall([A1,A2],ortho_pair(H,A1,A2),ListOrtho),
hf_rc_aux_aux2(ListOrtho,Aroms,IO,NewIO),

hf_rc_ortho_twist(ListOrtho,NewIO,PartOT1),
NewPart is PartOT + PartOT1,

findall([A11,A22],meta_pair(H,A11,A22),ListMeta),
hf_rc_aux_aux2(ListMeta,Aroms,IM,NewIM),

findall([A111,A222],para_pair(H,A111,A222),ListPara),
hf_rc_aux_aux2(ListPara,Aroms,IP,NewIP),

hf_rc_position_aux(T,Aroms,NewIO,O,NewIM,M,NewIP,P,NewPart,OT).

/*****************************************************************
Ortho Twist gets a correction for molecules with orthos that
go through different reactophores
*****************************************************************/
hf_rc_ortho_twist([],_,0).
hf_rc_ortho_twist([_|_],0.0,OT):-
hf_rc(ortho_twist,OT).
hf_rc_ortho_twist([_|_],_,0).

/*****************************************************************
aux2 deals w/ the chain length of atoms between the aroms-really adds
up the length of the compound...like a big ring correction...
and gets a value based upon that
*****************************************************************/
hf_rc_aux_aux2(List,Aroms,In,NewIn):-
```

```
hf_for_real(List,Aroms,Aroms,NewList),
flatten(NewList,FL),
delete_duplicates(FL,Atoms),
length(Atoms,LenA),
AA is LenA/2,
NewIn is In + AA.

/*******************************************************************
checks if these o,m,p actually go to the same reac..if not throw away
.._aux gets the list of Connects: [[arom1,arom2],[arom1,arom1]], etc
.._aux2 gets
*******************************************************************/
hf_for_real([],_,_,[]).
hf_for_real(List,_,Aroms,NewList):-
member(R,List),
member(R1,R),
pair(R1,R2),
reac_type(R2,_,aromatic),
delete(Aroms,R2,Rest),
hf_for_real_aux(List,Rest,[],Connects),
hf_for_real_aux2(List,Connects,[],NewList).

hf_for_real_aux([],_,A,A).
hf_for_real_aux([[A1,A2]|T],Rest,A,Out):-
all_paths(A1,_,PA1),
all_paths(A2,_,PA2),
hf_rc_aux2_aux(Rest,OtherOuts1),%designed and used below

hf_for_real_aux_aux(OtherOuts1,Rest,PA1,[],Connects1a),
hf_fraa22a(Connects1a,[],Connects1),
hf_for_real_aux_aux(OtherOuts1,Rest,PA2,[],Connects2a),
hf_fraa22a(Connects2a,[],Connects2),
hf_for_real_aux_aux2(Connects1,[],Shortest1),
hf_for_real_aux_aux2(Connects2,[],Shortest2),
Shortest1=[Ar1,_Num1],
Shortest2=[Ar2,_Num2],
NewCon=[Ar1,Ar2],
hf_for_real_aux(T,Rest,[NewCon|A],Out).


hf_fraa22a([],A,A).
hf_fraa22a([H|T],In,Connects1):-
append(H,In,RR),
hf_fraa22a(T,RR,Connects1).
```

```
hf_for_real_aux_aux2(Con,_A,Shortest1):-
hf_fraa2(Con,[],Short),
minimum(Short,MS),
findall([R,MS],member([R,MS],Con),[Shortest1]).

hf_fraa2([],A,A).
hf_fraa2([[_,L]|T],A,Short):-
hf_fraa2(T,[L|A],Short).

/*******************************************************************
Determines the distance to next arom
*******************************************************************/
hf_for_real_aux_aux([],_,_,A,A).
hf_for_real_aux_aux([H|T],[H2|T2],PA,New,Connects):-
findall([H2,Dist],(member([_Atom,[Atom2,[Dist|_],_,_]],PA),
   memberchk(Atom2,H)),ListA1),
hf_for_real_aux_aux(T,T2,PA,[ListA1|New],Connects).


/*******************************************************************
??? If C1=C2 then add H to the list of kept things
Else throw it away
*******************************************************************/
hf_for_real_aux2([],_,A,A).
hf_for_real_aux2([H|T],[[C1,C2]|T2],A,New):-
C1=C2,
hf_for_real_aux2(T,T2,[H|A],New).
hf_for_real_aux2([_|T],[[_C1,_C2]|T2],A,New):-
hf_for_real_aux2(T,T2,A,New).


/*******************************************************************
aux2 deals w/ the chain length of atoms between the aroms
aux2_aux gets the "Out" atoms from the other aromatic(s)
aux2_aux2 gets the shortest chain length from 1
aromatic to the other
*******************************************************************/
hf_rc_chain([],_,ListLens,Correct):-
hf_rc_aux2_ll(ListLens,UseListLens),
hf_rc_aux2_aux3(UseListLens,Correct).
hf_rc_chain([H|T],Aroms,In,NewCorrect):-
delete(Aroms,H,Rest),
hf_rc_aux2_aux(Rest,OtherOuts1),
flatten(OtherOuts1,OtherOuts),
all_paths(H,_,Paths),
findall(P,pair(H,P,_,_),Outs),
findall([Start,Reac,Len],(member(Start,Outs),
          member([Start,[Y,L|_]],Paths), %from all_paths
          memberchk(Y,OtherOuts),
  member(Reac,Aroms),
  reactophore_fact(Reac,_,Atoms,_,_),
  memberchk(Y,Atoms),
          minimum(L,Len)),Lengths),
hf_rc_aux2_aux2(Lengths,ListLens),
NewListLens = [ListLens|In],
hf_rc_chain(T,Aroms,NewListLens,NewCorrect).


/*****************************************************************
finds the ListLens that are to be used
*****************************************************************/
hf_rc_aux2_ll(ListLens,UseListLens):-
findall(X,(member(X,ListLens),
   X = [_,_,_,_]),List),
(List = [] -> ListLens = [UseListLens,_]
    ; flatten(List,UseListLens)
).

/*****************************************************************
finds the atom that the H-type is a part of
*****************************************************************/
hf_rc_aux2_aux([],[]).
hf_rc_aux2_aux([H|T],[OO|OtherOuts]) :-
findall(P,pair(H,P,_,_),OO),
hf_rc_aux2_aux(T,OtherOuts).

/*****************************************************************
finds all of the lengths that are in a list
*****************************************************************/
hf_rc_aux2_aux2([],[]).
hf_rc_aux2_aux2([[S,R,L]|T],[LL|ListLens]):-
findall(X,member([S,_,X],[[S,R,L]|T]),Lens),
minimum(Lens,LL),
delete([[S,R,L]|T],[S,_,_],Rest),
hf_rc_aux2_aux2(Rest,ListLens).

/*************************************************************
the new version sums up the bonds and gets a correction based on
the length (if 6 get special correction, otherwise get another)
```

```
??need to add something that counts the bonds in the middle too??
******************************************************************/
hf_rc_aux2_aux3(LL,Correct):-
sumlist(LL,Sum1),
(Sum1 = 6 -> hf_rc(6,X)
   ; hf_rc(pp,M),
     X is M*Sum1
),
Correct is X.

/*******************************************************************
Correction for molecules w/ two rings joined by chains of
atoms:

like ring_chain1: needs a position correct and a chain correct

have to fix _chain2_chk so it works for all types of ring_chains
not just 6-6 ring chains

GOT TO FILTER OUT NON-RING-CHAIN COMPOUNDS
*******************************************************************/
hf_ring_chain2(_,Filtered,Correct):-
length(Filtered,2),
hf_ring_chain2_chk(Filtered),
hf_rc2_position(Filtered,PositCorrect),
hf_rc2_chain(Filtered,Filtered,[],ChainLenCorrect),
Correct is PositCorrect + ChainLenCorrect.
hf_ring_chain2(_,_,0).

/********************************************************************
gets the correction for position that the ring is in.
I don't think it should even be in here if the rings are joined
ortho.
uses the same correction for meta and para for those that are
at positions 1,4 or greater, then use the para correction,
until a better model is proposed

Probably will fuck up if multiple chains coming off of this
*******************************************************************/
hf_rc2_position(Filtered,PositCorrect):-
findall([A,Ring],(member(Ring,Filtered),
  findall(X,(member(Y,Ring),
              reactophore_fact(_R,_,[X],_,_),
                 \+ memberchk(X,Ring),
                 bonded(X,_,Y)),A)),ListAs),
hf_rc2_position_aux(ListAs,0,Pos1,0,Pos2,0,Pos3,0,PosOther),
hf_rc(ortho,P1),
hf_rc(meta,P2),
hf_rc(para,P3),
PositCorrect is Pos1*P1 + Pos2*P2 + Pos3*P3 + PosOther*P3.


/******************************************************************
rc2_position_aux finds the number in each position off of
each ring
******************************************************************/
hf_rc2_position_aux([],O,O,M,M,P,P,PO,PO).
hf_rc2_position_aux([[A,R]|T],PartO,Ortho,PartM,Meta,
PartP,Para,PartOt,Other):-
hrc2a_aux(A,R,0,O,0,M,0,P,0,Ot),
NewPartO is PartO + O,
NewPartM is PartM + M,
NewPartP is PartP + P,
NewPartOt is PartOt + Ot,
hf_rc2_position_aux(T,NewPartO,Ortho,NewPartM,Meta,
NewPartP,Para,NewPartOt,Other).

hrc2a_aux([],_,O,O,M,M,P,P,Ot,Ot).
hrc2a_aux([H|T],Ring,IPO,LPO,IPM,LPM,IPP,LPP,IPOt,LPOt):-
reactophore_fact(Sub,_,[H],_,_),
hf_pseudo_ortho_subs(Ring,Sub,POs),
hf_pseudo_meta_subs(Ring,Sub,PMs),
hf_pseudo_para_subs(Ring,Sub,PPs),
hf_other_subs(Ring,Sub,POts),
length(POs,LPO1),
LPO2 is LPO1/2,
length(PMs,LPM1),
LPM2 is LPM1/2,
length(PPs,LPP1),
LPP2 is LPP1/2,
length(POts,LPOt1),
LPOt2 is LPOt1/2,
NewLPO is IPO + LPO2,
NewLPM is IPM + LPM2,
NewLPP is IPP + LPP2,
NewLPOt is IPOt + LPOt2,
hrc2a_aux(T,Ring,NewLPO,LPO,NewLPM,LPM,NewLPP,LPP,NewLPOt,LPOt).


/**********************************************************************
```

```
Returns the substituents that are OMP not real OMPs b/c not a phenyl ring
ps_meta and _para check for 6 membered ring, but ortho does not
b/c might need an ortho correction in other rings

!!!Got to fix so that will work for rings besides len6!!!
*************************************************************************/
hf_pseudo_ortho_subs(Ring,Sub,POrthos):-
pair(Sub,R1,A,_),
memberchk(A,Ring),
findall(Y,(pair(_,A,B,_),
   B \= R1,
   pair(_,B,X,Y),
   \+ memberchk(X,Ring)),POrthos).
hf_pseudo_meta_subs(Ring,Sub,PMetas):-
length(Ring,6),
pair(Sub,R1,A,_),
memberchk(A,Ring),
findall(Y,(pair(_,A,B,_),
   B \= R1,
   pair(_,B,C,_),
   A \= C,
   pair(_,C,X,Y),
   \+memberchk(X,Ring)),PMetas).
hf_pseudo_para_subs(Ring,Reac1,[Reac2]):-
length(Ring,6),
pair(Reac1,R1,A,_),
memberchk(A,Ring),
pair(_,A,B,_),
B \= R1,
pair(_,B,C,_),
A \= C,
pair(_,C,D,_),
B \= D,
pair(_,D,E,Reac2),
\+memberchk(E,Ring).


hf_other_subs(Ring,_Sub,POts):-
length(Ring,LR),
LR \= 6,
POts = [_].
/*** GOT TO WRITE SOME CODE FOR THIS***/
hf_other_subs(_,_,[]).
```

```
/*****************************************************************
rc2_chain deals w/ the chain length of atoms between the aroms
aux2_aux gets the "Out" atoms from the other aromatic(s)
aux2_aux2 gets the shortest chain length from 1
aromatic to the other

aux2_aux3(same call as in ring_chain1)
*****************************************************************/
hf_rc2_chain([],_,_,0).
hf_rc2_chain([H|_],Filtered,_,Correct):-
delete(Filtered,H,Rest),
hf_rc2_aux2_aux(Rest,OtherOuts1),
flatten(OtherOuts1,OtherOuts),
hf_rc2_aux2_aux([H],Outs),
flatten(Outs,Outs1),
hf_rc2aa(Outs1,OtherOuts,Filtered,Lengths),
hf_rc_aux2_aux3(Lengths,Correct).

/****************************************************************
Finds the Lengths between the Outs
****************************************************************/
hf_rc2aa([],_,_,[]).
hf_rc2aa([H|T],OtherOuts,Filtered,[Len1|Rest]):-
findall(Ring,(member(Ring,Filtered),
      memberchk(H,Ring)),[Ring]),
reactophore_fact(Sub,_,[H],_,_),
all_paths(Sub,_,Paths),
findall(Len,(member([H,[Y,[0]|End]],Paths), %from all_paths
     memberchk(Y,OtherOuts),
     \+ memberchk(Y,Ring),
     \+ memberchk([_,_],End),
     delete_duplicates([Y,[0]|End],Short),
     delete(Short,[_|_],Path),
     length(Path,Len)),[Len]),
Len1 is Len-1,
hf_rc2aa(T,OtherOuts,Filtered,Rest).

/****************************************************************
Finds the Outs
****************************************************************/
hf_rc2_aux2_aux([],[]).
hf_rc2_aux2_aux([Ring|T],[OO|OtherOuts]) :-
```

```
findall(A,(member(A,Ring),
   pair(_,A,X,_),
   \+memberchk(X,Ring)),OO),
hf_rc2_aux2_aux(T,OtherOuts).

/****************************************************************
Checks that the molecule is actually a ring_chain
*****************************************************************/
hf_ring_chain2_chk([]).
hf_ring_chain2_chk([H|T]):-
length(H,6),
findall(A,(member(X,H),
   pair(_,X,A,_),
   \+memberchk(A,H)),[_,_]),
hf_ring_chain2_chk(T).

/******************************************************************
New hf_ring_correct calls a module in hf_ring2.pro (
hf_ring_structure_correct)

*******************************************************************/

hf_ring_correct([],All,RAs,Part,Value):-
hf_ring_structure_correct(RAs,All,Val),
Value is Part + Val.
hf_ring_correct([H|T],All,RAs,Part,Val):-
length(H,Len),
get_hf_ring(Len,V),

hf_ring_deform(Len,H,All,RingDeform),

hf_ring_aromatic(Len,H,All,RingBridge),

NewPart is Part + V + RingBridge + RingDeform,
hf_ring_correct(T,All,RAs,NewPart,Val).

/******************************************************************
Correction for the deformation rings undergo for having
-sp2 atoms a part of their structure that can't be accounted
for in their appropriate section b/c this describes how
  the RING is deformed by their being a part of it

-have to have a correction for each Len

 (THIS SHOULD PROBABLY SUBSUME hf_ring_aromatic)
```

```
*****************************************************************/
hf_ring_deform(Len,Ring,_All,Value):-
findall(X,(reactophore_fact(X,e,[A,B],_,_),
   intersects([A,B],Ring,[_])),RingExos),
length(RingExos,NumExos),
findall(X,(reactophore_fact(X,e,[A,B],_,_),
   intersects([A,B],Ring,[_,_])),RingEndos),
length(RingEndos,NumEndos),
findall(X,(reactophore_fact(X,_,Atoms,_,_),
   reac_type(X,_,aromatic),
   intersects(Atoms,Ring,[_,_])),RingAroms),
length(RingAroms,NumAroms),
hf_ring_deform_aux(Len,NumExos,RingExos,NumEndos,
RingEndos,NumAroms,RingAroms,Ring,Value).

hf_ring_deform_aux(_,0,_,0,_,0,_,_,0).
hf_ring_deform_aux(3,_NumExos,_RingExos,_NumEndos,
_RingEndos,_NumAroms,_RingAroms,_Ring,0).
hf_ring_deform_aux(4,_NumExos,_RingExos,_NumEndos,
_RingEndos,_NumAroms,_RingAroms,_Ring,0).
hf_ring_deform_aux(5,1,_,2,_,0,_,_,Value):-
hf_ring_deform(5,Value).
hf_ring_deform_aux(6,_NumExos,RingExos,2,RingEndos,0,_,_Ring,Value):-
member(Endo1,RingEndos),
member(Exo1,RingExos),
pair(Endo1,Exo1),
member(Endo2,RingEndos),
Endo2 \= Endo1,
pair(Endo2,Exo1),
hf_ring_deform(6,Value).
hf_ring_deform_aux(7,NumExos,_,3,_,0,_,_,Value):-
NumExos < 2,
Value is 0.
hf_ring_deform_aux(8,_NumExos,_RingExos,_NumEndos,
_RingEndos,_NumAroms,_RingAroms,_Ring,0).
hf_ring_deform_aux(_Len,_,_,_,_,_,_,_,0).
% General rule for cases not thought of yet




/**************************************************************************
```

```
Correction for Rings that share a side with an
aromatic. (RSizeVal*BridgeNum)

ring_aromatic_aux2 - gets a correction for a db in a 6 membered
     ring being next to an aromatic ~18kJ.

Additional correction for multiple aroms on a ring
Value (from hf_multiple_arom).
***************************************************************************/
hf_ring_aromatic(Len,R,All,Value) :-
findall(XX,(member(XX,R),
    atom_spec(_,XX,_,_,_)),List),
hf_ring_aromatic_aux(List,Len,R,All,0,Val1),
hf_ring_aromatic_aux2(List,Len,Val2),
hf_multiple_arom(List,R,All,0,Val3),
Value is Val1 + Val2 - Val3.

hf_ring_aromatic_aux([],_,_,_,Out,Out).
hf_ring_aromatic_aux([H|T],Len,R,All,In,Out):-
findall(X,(member(X,All),
   memberchk(H,X)),Rings),
findall(XX,(bonded(H,_,XX),
            \+ memberchk(XX,T),
    member(RR,All),
    RR \= R,
    memberchk(XX,RR),
    member(RRR,All),
    RRR \= R,
    RRR \= RR,
    memberchk(XX,RRR)),ListSSS),
(Rings = [_] -> true
      ; (ListSSS = [] -> true
       ; fail
)
),
(Len < 8 -> arom_ring_side(Len,RSizeVal)
  ; arom_ring_side(other,RSizeVal)
),
NewIn is In + RSizeVal,
hf_ring_aromatic_aux(T,Len,R,All,NewIn,Out).
hf_ring_aromatic_aux([_|T],Len,R,All,In,Out):-
hf_ring_aromatic_aux(T,Len,R,All,In,Out).


hf_ring_aromatic_aux2(List,5,Out):-  %this only fires for exo-oeth?
```

```
length(List,LL),
LL > 2,
findall(X,(member(X,List),
   bonded(X,_,Y),
   reac_type(Y,ethylenic),
   member(Z,List),
   Z \= X,
   bonded(Y,_,Z),
   reactophore_fact(A,_,C,_,_),
   memberchk(Y,C),
   n_o_s_type(A,oethylenic)),[_|_]),
arom_ring_side(db5,Out).

/****** this fires for every 5-ethy (including CAS: 208968)
length(List,LL),
LL > 2,
findall(X,(member(X,List),
   bonded(X,_,Y),
   reac_type(Y,ethylenic)),[_|_]),
arom_ring_side(db5,Out).
*******/
hf_ring_aromatic_aux2(List,6,Out):-
findall(X,(member(X,List),
   bonded(X,_,Y),
   reac_type(Y,ethylenic)),[_|_]),
arom_ring_side(db,Out).
hf_ring_aromatic_aux2(_,_,0).




hf_multiple_arom([],_,_,Out,Out).
hf_multiple_arom([H|T],R,All,In,Value) :-
findall(X,(member(X,All),
   memberchk(H,X)),Rings),
findall(XX,(bonded(H,_,XX),
            \+ memberchk(XX,T),
    member(RR,All),
    RR \= R,
    memberchk(XX,RR),
    member(RRR,All),
    RRR \= R,
    RRR \= RR,
    memberchk(XX,RRR)),ListSSS),
(Rings = [_] -> true
      ; (ListSSS = [] -> true
       ; fail
)
),
findall(X,(member(A,R),
   atom_spec(_,A,_,_,_),
   reactophore_fact(X,_,L,_,_),
   memberchk(A,L)),List),
delete_duplicates(List,NewList),
length(NewList,Len),
Len > 1,
length(R,LenR),
arom_ring_side(maromA,A),
arom_ring_side(maromB,B),
NewIn is In + A*LenR*LenR + B*LenR,
hf_multiple_arom(T,R,All,NewIn,Value).
hf_multiple_arom([_|T],R,All,In,Value) :-
hf_multiple_arom(T,R,All,In,Value).

/*****************************************************************
Correction that depends on the size of the ring being
looked at:
if the size is < 6 looks up a correction in the database
if it's >= 6 then uses a 'morse potential' to calculate
the value of the correction.
A Morse potential has the form:
Val = D*(1-e^(-B*(Len-6)))^2
where D is the height of the function, B is where the function
rolls over, Len is the point sitting on.

may need to train D.  right now D=52.23 b/c that is decane's diff

Looks like Morse potential until size = 10
At size = 11 have an exponential decay
******************************************************************/
get_hf_ring(Len,Val) :-
Len < 6,
hf_ring(Len,Val),
!.
get_hf_ring(Len,Val) :-
6 =< Len,
Len =< 10,
hf_ring(mp_b,B),
hf_ring(mp_d,D),
X is (1-exp(-B*(Len-6))),%Morse potential
```

```
pow(X,2,NX),
Val is D*NX.
get_hf_ring(Len,Val) :-
Len > 10,
hf_ring(ed_a,A),
hf_ring(ed_b,B),
Val is A*exp(-B*Len).

/******************************************************************
    Gets the basic correction for chains and aromatic compounds
HFStart is the correction each reac_type has: methyl=-74.6, etc

Correction is the amount of correction based on:
steric, connectivity, resonance, etc
******************************************************************/
reac_hf(reactophore_fact(R1,_,_,_,_),HF) :-
reac_type(R1,_,ReacType),
get_hf_contrib(ReacType,R1,HFStart),
hf_data(basic,R1,HFStart),

hf_correct(ReacType,R1,Resonance,Correction),

HF is HFStart + Correction + Resonance.
%% HF is HFStart + Correction.


/******************************************************************
Gets the charge distributed out of the reactophore
makes sure that only aroms,ethys,and acetys go through
the charge_distributor routine...everything else gets a 0.

the aux routine gets the Qin distributed out of the
reactophore,  [H|T] are list of atoms that are 'outs'
of the reactophore.

get's normalized by taking away 1 from the Qin depending
on the number of outs from the reactophore...
for 2reacs -1,
for 3reacs -2, etc
******************************************************************/
hf_get_chargeout(Reac,TadChains,ChargeOut):-
reac_type(Reac,_,RType),
memberchk(RType,[aromatic,ethylenic,acetylenic]),
findall(A,(pair(Reac,A,B,_),
   \+ atom_type(B,q)),Outs),
hf_get_chargeout_aux(Outs,Reac,TadChains,0,ChargeOutx),
length(Outs,LenO),
ChargeOut is 1-1/(ChargeOutx-(LenO-1)).
hf_get_chargeout(_,_,0).


hf_get_chargeout_aux([],_,_,Part,Part).
hf_get_chargeout_aux([H|T],Reac,TadChains,Part,ChargeOut):-
        charge_distributor(Reac,H,gamacalc,TadChains,true,_,Qin),
NewPart is Part + Qin,
hf_get_chargeout_aux(T,Reac,TadChains,NewPart,ChargeOut).

/************************************************************************
Contribution to Hf from the reactophore types:
methyl, ethylenic, acetylenic, aromatics(calls oa_aux)
1. not one of the following types get a 0
2. for aromatics
%%% 3. for [methyl-groups) halogens,oeth,seth?
4. for the other groups


!!need to look at the stuff that deals w/ halogens eventually
************************************************************************/
get_hf_contrib(ReacType,_,0) :-
       \+ memberchk(ReacType,[methyl,aromatic,ethylenic,acetylenic, %carbons
      oh,co2h,oeth,    %oxygens
%%%       nr2,cn,    %nitrogens
      f,cl,br,i     %halogens
%%%       sh,    %sulfurs
%%%       pr2     %phosphorus
     ]).
get_hf_contrib(aromatic,R1,HFStart) :-
oa_aux(reactophore_fact(R1,_,_,_,_),HFStart).
get_hf_contrib(ReacType,R1,HFStart) :-
memberchk(ReacType,[f,cl,br,i]),
hf_halo_contrib(ReacType,R1,HFStart).
get_hf_contrib(ReacType,_,HFStart) :-
hf_contrib(ReacType,HFStart).

/********************************************************************
takes a halogen and sees what it's attached to, gets the
basic contribution based on this
********************************************************************/
hf_halo_contrib(ReacType,R1,HFStart):-
pair(R1,Sub),
reac_type(Sub,_,Type),
hf_contrib(ReacType-Type,HFStart).
hf_halo_contrib(_,_,0).



/*******************************************************************
correction for number of non-carbons (halos for now 12/19/01)
on molecule
*******************************************************************/
hf_ref_contrib([],HF,HF).
hf_ref_contrib([H|T],Part,HF) :-
hf_contrib(H,HF0),
NewPart is HF0 + Part,
hf_ref_contrib(T,NewPart,HF).
hf_ref_contrib([_|T],NewPart,HF) :-
hf_ref_contrib(T,NewPart,HF).
/*******************************************************************
Correction for when there are multiple noncarbons (Subs)
branching off of the carbon with the 'noncarbon' attached
to it.   eg. CBr4
  hf_multi_ref is the reference point for certain number,
     doesn't care what the Subs are.
*******************************************************************/
hf_multi_reduction(Subs,HF0,HF) :-
length(Subs,L),
L > 1,
!,
hf_multi_charge_correction(L,Subs,0,Add),
hf_multi_ref(L,Mult),
hf_multi_ref(const,Constant),%why do we need this constant?
HF is HF0 * Mult + Constant + Add.
hf_multi_reduction(_,HF,HF).


/*******************************************************************
Correction for the type of 'noncarbons'
  hf_multi_ref is correction for what Subs are attached
*******************************************************************/
hf_multi_charge_correction(_,[],Add,Add).
hf_multi_charge_correction(L,[H|T],Part,Add) :-
substituent_type(H,_,Type),
pka_s_f(Type,Chi),%electronegativity
hf_multi_charge(L,Mult),
NewPart is Part + Mult * (Chi - 2.3),
        hf_multi_charge_correction(L,T,NewPart,Add).
hf_multi_charge_correction(L,[_|T],NewPart,Add) :-
hf_multi_charge_correction(L,T,NewPart,Add).




/**********************************************************************
Correction to Hf for each of the reactophore types

inside of ethylenic, have to check if this ethylenic is really
an neth,oeth,or seth, or just a regular ethylene.

**********************************************************************/
hf_correct(methyl,Sub,0,Value):-
findall(X,pair(Sub,X),Subs),
        hf_branch(methyl,Sub,_,Subs,Val),
hf_halo(methyl,Sub,Halo),
Value is Val + Halo,
        !.
hf_correct(ethylenic,Sub,ValueR,ValueB):-%for neth_oeth_seth
neth_oeth_seth(Sub),%n_o_s? yes -> keep going
n_o_s_type(Sub,Type),
hf_branch(Type,Sub,_,_,ValueB1),%steric and strain

hf_pi_subs(Sub,ListPi),

hf_dipole_repulsion(ListPi,Type,0,ValueB2),
hf_data(dipole,Sub,ValueB2),

ValueB is ValueB1 + ValueB2,
hf_resonance(ListPi,_,Sub,Type,0,ValueR),
        !.
hf_correct(ethylenic,Sub,ValueR,ValueB):-%for regular ethylenic
\+ neth_oeth_seth(Sub),
n_o_s_type(Sub,Type),
hf_branch(Type,Sub,_,_,ValueB),%steric and strain

hf_pi_subs(Sub,ListPi),

hf_resonance(ListPi,_,Sub,Type,0,ValueR),
        !.
hf_correct(acetylenic,Sub,ValueR,ValueB):-
findall(X,pair(Sub,X),Subs),
hf_branch(acetylenic,Sub,_,Subs,Val),
hf_halo(acetylenic,Sub,Halo),

ValueB is Val + Halo,


hf_pi_subs(Sub,ListPi),

hf_resonance(ListPi,Subs,Sub,acetylenic,0,ValueR),
!.
hf_correct(aromatic,Sub,ValueR,ValueB) :-
findall(X,pair(Sub,X),Subs),
hf_branch(aromatic,Sub,_,Subs,ValueB),
hf_pi_subs(Sub,ListPi),
hf_resonance(ListPi,Subs,Sub,aromatic,0,ValueR),
!.
hf_correct(oh,Sub,0,Value):-
findall(X,pair(Sub,X),Subs),
hf_branch(oh,Sub,_,Subs,Value),
        !.
hf_correct(co2h,Sub,0,Value):-
findall(X,pair(Sub,X),Subs),
hf_branch(co2h,Sub,_,Subs,Value),
        !.
hf_correct(SubType,Sub,0,ValueB) :-
fail,
findall(X,pair(Sub,X),Subs),
hf_branch(SubType,Sub,_,Subs,ValueB),
!.
hf_correct(_,_,0,0).

/***********************************************************************
This is for when a halogen is attached to a substituent
***********************************************************************/
hf_halo(Type,Sub,Halo):-
hf_halo_aux3(Sub,Hals,Halos),
Halos \= [],
hf_halo_aux(Type,Halos,Sub,Hal),%interaction between halos on same atom
hf_halo_aux2(Type,Halos,Sub,Xi),
%interaction between halos and adjacent atoms
hf_halo_per(Type,Hals,Halos,Sub,Per),
Halo is Hal + Xi + Per,

        hf_data(connectivity,Sub,Halo).
hf_halo(_,_,0).
```

```
/**********************************************************************
Finds the Halos off a given Sub
**********************************************************************/
hf_halo_aux3(Sub,Hals,Halos):-
findall(X,(pair(Sub,Hal),
   reac_type(Hal,_,X),
   memberchk(X,[f,cl,br,i])),Halos),
findall(Hal,(pair(Sub,Hal),
      reac_type(Hal,_,X),
      memberchk(X,[f,cl,br,i])),Hals).

/**********************************************************************
This code will see what each Sub is hooked to and get the appropriate
correction
**********************************************************************/
hf_halo_aux2(methyl,Halos,Sub,Xi):-
findall(OSub,(pair(Sub,OSub),
      reac_type(OSub,_,OType),
      \+ memberchk(OType,[f,cl,br,i])),OSubs),
hf_halo_aux2_aux1(OSubs,Halos,0,Xi).
hf_halo_aux2(ethylenic,_,Sub,Xi):-
reactophore_fact(Sub,_,[A,B],_,Atoms),
hf_halo_aux2_ethy1(Sub,A,[_|_]),
%make sure at least one halo on each side of ethy
hf_halo_aux2_ethy1(Sub,B,[_|_]),
hf_halo_aux2_ethy(Atoms,Xi).
hf_halo_aux2(_,_,_,0).

/**********************************************************************
Make sure there are Halos off off a Sub  (maybe can combine w/ hf_halo_aux3)
**********************************************************************/
hf_halo_aux2_ethy1(Sub,A,Halos):-
findall(Type,(pair(Sub,A,_,X),
      reac_type(X,_,Type),
      memberchk(Type,[f,cl,br,i])),Halos).

/**********************************************************************
Get the "cis" interaction and then if there is no "cis" interaction
get other correction (only necessary for fluorine)
**********************************************************************/
hf_halo_aux2_ethy([[A1,A2],[B1,B2]],Xi1):-
hf_halo_aux2_ethy_aux(A1,B2,Val1a),
hf_halo_aux2_ethy_aux(A2,B1,Val1b),
hf_halo_aux2_ethy_aux2(Val1a,Val1b,[A1,A2,B1,B2],Val2),
Xi1 is Val1a + Val1b + Val2.
```

```
%%%%
hf_halo_aux2_ethy_aux2(Val1a,Val1b,_,0):-
(Val1a \= 0 ; Val1b \= 0).
hf_halo_aux2_ethy_aux2(_,_,[A1,A2,B1,B2],Val2):-
hf_halo_aux2_ethy_aux2_aux([A1,B1],Val2a),
hf_halo_aux2_ethy_aux2_aux([A2,B2],Val2b),
Val2 is Val2a + Val2b.

%%%%
hf_halo_aux2_ethy_aux2_aux([[A],[B]],Val):-
%% only for fluorine (Cl seems to work just fine)
hf_halo_get_halo(A,f),
hf_halo_get_halo(B,f),
hf_halo_get_key(f,f,Key),
hf_halo_ethy3(Key,Val).
hf_halo_aux2_ethy_aux2_aux(_,0).

%%%%
hf_halo_aux2_ethy_aux([A],[B],Val1):-
hf_halo_get_halo(A,AHal),
hf_halo_get_halo(B,BHal),
hf_halo_get_key(AHal,BHal,Key),
hf_halo_ethy2(Key,Val1).
hf_halo_aux2_ethy_aux(_,_,0).

/***********************************************************
Get the correction for the halos off of each Sub
***********************************************************/
hf_halo_aux2_aux1([],_,Xi,Xi).
hf_halo_aux2_aux1([OSub|T],Halos,Part,Xi):-
hf_halo_aux3(OSub,_,OHalos),
reac_type(OSub,_,OType),
hf_halo_aux2_aux(OType,OSub,OHalos,Halos,Val),
NewPart is Part + Val,
hf_halo_aux2_aux1(T,Halos,NewPart,Xi).

hf_halo_aux2_aux(methyl,_,[],Halos,Xi):-
%for Halos attached to pure methyl
length(Halos,Len),
delete_duplicates(Halos,Hals),
concatenate_atomlist_to_atom(Hals,Key1),
concatenate_atomlist_to_atom([Key1-Len],Key),
hf_halo_xi(Key,Xi).
hf_halo_aux2_aux(methyl,_,[],_,Xi):-
```

```
Xi is error.
hf_halo_aux2_aux(ethylenic,_,[],[f,f,f],Xi):-
%for when a methyl is attached to an ethy
hf_halo_xi(f3-e,Xi).
hf_halo_aux2_aux(aromatic,_,_,[f,f,f],Xi):-
%for when a methyl is attached to an arom
hf_halo_xi(f3-arom,Xi).

hf_halo_aux2_aux(methyl,_,OHalos,Halos,Xi):-
%for Halos attached to methyl w/ multiHalos
hf_halo_aux2_aux_aux(Halos,OHalos,0,Xi).
hf_halo_aux2_aux(_,_,_,_,0).

/***************************************************************
for calculating the interaction between Halos on adjacent atoms
***************************************************************/
hf_halo_aux2_aux_aux([],_,Xi,Xi).
hf_halo_aux2_aux_aux([H|T],OHalos,Part,Xi):-
hf_halo_aux2_aux_aux_aux(OHalos,H,0,PXi),
NewPart is Part + PXi,
hf_halo_aux2_aux_aux(T,OHalos,NewPart,Xi).

hf_halo_aux2_aux_aux_aux([],_,PXi,PXi).
hf_halo_aux2_aux_aux_aux([H|T],Hal,Part,PXi):-
hf_halo_get_key(H,Hal,Key),
hf_halo_xi(Key,V),
NewPart is Part + V,
hf_halo_aux2_aux_aux_aux(T,Hal,NewPart,PXi).

/***************************************************************
for calculating the perfluoro interaction
***************************************************************/
hf_halo_per(methyl,_,Halos,Sub,Per):-
hf_halo_per_common(Halos,Sub,Len,OSubs,LenOS),
(Len =:= 3 -> true
    ; Len =:= 2, LenOS =:= 2 -> true
          ; Len =:= 1, LenOS =:= 3 -> true
; fail
),
hf_halo_per_aux(methyl,OSubs,Sub,Len,0,Per).
hf_halo_per(ethylenic,Hals,Halos,Sub,Per):-
hf_halo_per_common(Halos,Sub,Len,OSubs,LenOS),
(Len =:= 3 -> true
    ; Len =:= 2,
      LenOS =:= 2,
      Hals = [A,B],
      pair(A,_,C1,S),
      pair(B,_,C2,S),
      C1 \= C2,
      reactophore_fact(S,_,_,_,_) -> true
           ; fail
),
hf_halo_per_aux(ethylenic,OSubs,Sub,_,0,Per).
hf_halo_per(_,_,_,_,0).

%%%
hf_halo_per_common(Halos,Sub,Len,OSubs,LenOS):-
delete_duplicates(Halos,[f]),
length(Halos,Len),
findall(OSub,(pair(Sub,OSub),
      reac_type(OSub,_,OType),
      \+ memberchk(OType,[f,cl,br,i])),OSubs),
length(OSubs,LenOS).

%%%
hf_halo_per_aux(_,[],_,_,Per,Per).
hf_halo_per_aux(methyl,[OSub|T],Sub,NFS,Part,Per):-
hf_halo_per_common2(OSub,Sub,_),
hf_halo(per-NFS,Val),
NewPart is Part + Val,
hf_halo_per_aux(methyl,T,Sub,NFS,NewPart,Per).
hf_halo_per_aux(ethylenic,[OSub|T],Sub,_,Part,Per):-
hf_halo_per_common2(OSub,Sub,Len),
hf_halo_ethy1(per-Len,Val),
NewPart is Part + Val,
hf_halo_per_aux(ethylenic,T,Sub,_,NewPart,Per).
hf_halo_per_aux(_,[_|T],Sub,NFS,Part,Per):-
hf_halo_per_aux(_,T,Sub,NFS,Part,Per).

%%%
hf_halo_per_common2(OSub,Sub,Len):-
hf_halo_aux3(OSub,_,OHalos),
delete_duplicates(OHalos,[f]),
length(OHalos,Len),
findall(OOSub,(pair(OSub,OOSub),
       OOSub \= Sub,
       reac_type(OOSub,_,OOType),
       \+ memberchk(OOType,[f,cl,br,i])),OOSubs),
length(OOSubs,LenOOS),
(Len =:= 3 -> true
    ; Len =:= 2, LenOOS =:= 1  -> true
  ; fail
).


/*********************************************************************
%%% THIS Determines the xi mult, the real way to do it, i'm going to hack
%%% something together and then see if we can come back and do it right when
%%% get more data

hf_halo_aux2(methyl,Sub,Halos,Halo):-
fail,
hf_halo_aux_out(Sub,Outs),
length(Outs,Len),
Len > 0,
hf_halo_aux_aux(Outs,Halos,Sub,[],Halo).
hf_halo_aux2(_,_,_,0).

%%%%
hf_halo_aux_out(Sub,Outs):-
findall(Out,(pair(Sub,Out),
     reac_type(Out,_,X),
     \+ memberchk(X,[f,cl,br,i])),Outs).

%%%%
hf_halo_aux_aux([],_,_,Xis,Halo):-
sumlist(Xis,Halo).
hf_halo_aux_aux([Out|Rest],Halos,Sub,Part,Halo):-
hf_get_effective_xi(Halos,Sub,Out,XiC1),

findall(O2,(pair(Sub,O2),
    reac_type(O2,_,X),
    \+ memberchk(X,[methyl])),O2s),
hf_get_effective_xi(O2s,Out,Sub,XiC2),
P1 is XiC2-XiC1,
abs(P1,P2),

hf_halo(f,Q),

XX is Q*(P2 - 4.6),
abs(XX,XXX),
hf_halo_aux_aux(Rest,Halos,Sub,[XXX|Part],Halo).


%%Gets the effective Chi for subs
```

```
%%for pure C it's 2.3

%%Subs are the Substiuents off of C1,
%%C1 is the reac_center
%%C2 is the other reac
%%XiC1 is the Xi getting

hf_get_effective_xi(Subs,C1,C2,XiC1):-
        reac_type(C1,_,C1Type),
        pka_s_f(C1Type,StartXi),
        reac_type(C2,_,C2Type),
        sigma1(Subs,C1,StartXi,C2,C2Type,0.295,0,sigma,0.911766,[],XiC1).


test_xi(X):-
        Smiles = 'FCCC',
        execute(prepare_for_property(nul,Smiles),ResTop),
        ResTop = true,
        !,

        findall(X,(pair(m1,X),
                   \+ reac_type(X,_,methyl)),Subs),

        hf_get_effective_xi(Subs,m1,m2,X).

*****************************************************************/

/*****************************************************************
Gets the intra-atom interaction between Halos
*****************************************************************/
hf_halo_aux(methyl,Halos,_,0):-
length(Halos,NumHalos),
NumHalos =< 1.
hf_halo_aux(methyl,Halos,_,Halo):-%mutiple Halos on same sub
hf_halo_part(Halos,f,FPart),
hf_halo_part(Halos,cl,ClPart),
hf_halo_part(Halos,br,BrPart),
hf_halo_part(Halos,i,IPart),
Part1 is FPart + ClPart + BrPart + IPart,
hf_halo_aux_aux(Halos,Halos,0,Part2),
%gets interaction between each halo and the others
Halo is Part1 + Part2.
hf_halo_aux(ethylenic,_,Sub,Halo):-
reactophore_fact(Sub,_,_,_,[AA,BB]),
hf_halo_aux_ethy(AA,P1),
hf_halo_aux_ethy(BB,P2),
Halo is P1 + P2.

/*****************************************************************
Gets interaction between Halos on same atom in an ethy
*****************************************************************/
hf_halo_aux_ethy([[],[]],0).
hf_halo_aux_ethy([[_],[]],0).
hf_halo_aux_ethy([[],[_]],0).
hf_halo_aux_ethy([[A1],[A2]],P1):-
hf_halo_get_halo(A1,H1),
hf_halo_get_halo(A2,H2),
hf_halo_get_key(H1,H2,Key),
hf_halo_ethy1(Key,P1).
hf_halo_aux_ethy(_,0).

/********************************
Gets the "Key" from two inputs
********************************/
hf_halo_get_key(H1,H1,H1-H1).
hf_halo_get_key(H1,H2,Key):-
sort([H1,H2],[A,B]),
Key = A-B.

/********************************
Takes and atom and see what halogen
is paired to it (used by ethylenics)
********************************/
hf_halo_get_halo(Atom,Halo):-
reactophore_fact(Sub1,_,At1,_,_),
memberchk(Atom,At1),
reac_type(Sub1,_,Halo),
memberchk(Halo,[f,cl,br,i]).

/*****************************************************************
gets correction for each halogen interaction with the others
if same type don't get interaction, just recall
*****************************************************************/
hf_halo_aux_aux([],_,Out,Out).
hf_halo_aux_aux([H|T],Halos,Part,Part2):-
hf_halo_aux_aux_aux(Halos,H,0,Val),
NewPart is Part + Val,
hf_halo_aux_aux(T,Halos,NewPart,Part2).
```

```
hf_halo_aux_aux_aux([],_,Out,Out).
hf_halo_aux_aux_aux([H|T],H,Part,Val):-
hf_halo_aux_aux_aux(T,H,Part,Val).
hf_halo_aux_aux_aux([H2|T],H,Part,Val):-
hf_halo_get_key(H2,H,Key),
hf_halo(Key,V),
NewPart is Part + V,
hf_halo_aux_aux_aux(T,H,NewPart,Val).

/**************************************************************
Gets the correction for a Type of halogen
(polynomial correction)
**************************************************************/
hf_halo_part(Halos,Type,Part):-
hf_num_halos(Halos,Type,NumX),
concatenate_atomlist_to_atom([Type-a],KeyA),
concatenate_atomlist_to_atom([Type-b],KeyB),
hf_halo(KeyA,VA),
hf_halo(KeyB,VB),
Part is VA*NumX*NumX+VB*NumX.

hf_num_halos(Halos,Type,Num):-
findall(Type,member(Type,Halos),Xs),
length(Xs,Num).


/********************************************************************
Code is written to account for the raising of energy by ~25kJ
when two oeths are joined next to each other
********************************************************************/
hf_dipole_repulsion([],_,Out,Out).
hf_dipole_repulsion([H|T],Type,Part,ValueB2):-
n_o_s_type(H,HType),
hf_dipole(HType,Val),
NewPart is Part + Val,
hf_dipole_repulsion(T,Type,NewPart,ValueB2).
hf_dipole_repulsion([_|T],Type,Part,ValueB2):-
hf_dipole_repulsion(T,Type,Part,ValueB2).

/********************************************************************
Correction for resonance in chains and chains-rings
Sub is whatever substituent you are on (eg:  e2) ListPi are the
other ethylenic or aromatic compounds that are hooked to it
********************************************************************/
hf_resonance([_|_],_Subs,Sub,Type,_Part,Resonance) :-
```

```
allchains(Chains),
        hf_fix_chain(Chains,Sub,TadChain),
        delete_duplicates(TadChain,TadChains1),
hf_fix_chain2(TadChains1,TadChains),

hf_get_chargeout(Sub,TadChains,ChargeOut),
hf_resonance(Type,ResVal),

Resonance is ChargeOut*ResVal*(-1.0),
        hf_data(resonance,Sub,Resonance).
hf_resonance(_,_,Sub,_,_,0):-
        hf_data(resonance,Sub,0).


/*******************************************************
deletes acetylenics from chain because no charge
should get distributed into them
*******************************************************/
hf_fix_chain2([],[]).
hf_fix_chain2([H|T],[H|TadChains]):-
H = [_],
!,
hf_fix_chain2(T,TadChains).
hf_fix_chain2([H|T],[H|TadChains]):-
H = [_,Name|_],
\+ reac_type(Name,_,acetylenic),
!,
hf_fix_chain2(T,TadChains).
hf_fix_chain2([_|T],TadChains):-
hf_fix_chain2(T,TadChains).



/*******************************************************
arranges the Chains so that gets the reactophores to
the left and right of the reactophore you are sitting
on
*******************************************************/
hf_fix_chain([],_,[]).
hf_fix_chain([H|T],Sub,Chains) :-
\+ memberchk(Sub,H),
hf_fix_chain(T,Sub,Chains).
hf_fix_chain([H|T],Sub,[[Sub|RLeft],[Sub|Right]|Chains]) :-
memberchk(Sub,H),
chop_at_item(H,Sub,Left,Right),
reverse(Left,RLeft),
hf_fix_chain(T,Sub,Chains).

/*********************************************************************
 Correction for size and number of methyl,ethylenic,acetylenic,
 and aromatic groups:
Len is number of substituent groups,
Effective Size is the size of these groups

   hf_perri gets correction for groups that are perri

 get's oh groups connectivity correction too.

**********************************************************************/
hf_branch(methyl,Sub,_,Subs,Value) :-
length(Subs,Len),
Len =< 4,

hf_branch_connectivity(methyl,Len,Sub,Subs,Connectivity),

hf_get_sub_size(Sub,Subs,Sizes),%capped at size of 2-t-butyls
%% get_sub_size(Sub,Subs,Sizes),
sumlist(Sizes,Size),
ESize1 is Size - Len * 0.05,
%%      minimum([ESize1,0.735],ESize),
ESize is ESize1,

maximum([0,ESize], EffectiveSize),
hf_steric_methyl(Len, Hfs),
        Steric is EffectiveSize * Hfs,
        hf_data(steric,Sub,Steric),

        Value is Connectivity + Steric.

hf_branch(ethylenic,Sub,_,_,Value) :-
hf_cis_trans(Sub,ethylenic,Value). %really deals w/ atoms around ethy
hf_branch(oethylenic,Sub,_,_,Value) :-
hf_cis_trans(Sub,oethylenic,Value).
hf_branch(nethylenic,Sub,_,_,Value) :-
hf_cis_trans(Sub,nethylenic,Value).
hf_branch(sethylenic,Sub,_,_,Value) :-
hf_cis_trans(Sub,sethylenic,Value).

hf_branch(acetylenic,Sub,_,Subs,Value) :-
length(Subs,Len),
hf_branch_acetylenic(a,A),
hf_branch_acetylenic(b,B),
Connectivity is ((A*Len*Len)+(B*Len)),
        hf_data(connectivity,Sub,Connectivity),

hf_get_sub_size(Sub,Subs,Sizes),
%% get_sub_size(Sub,Subs,Sizes),
sumlist(Sizes,Size),
ESize is Size - Len * 0.05,
maximum([0,ESize], EffectiveSize),
hf_steric_acetylenic(ace, Hfs),
        Steric is EffectiveSize * Hfs,
        hf_data(steric,Sub,Steric),
hf_endo_ace(Sub,Endo),
hf_data(basic_endo,Sub,Endo),

        Value is Connectivity + Steric + Endo.
hf_branch(aromatic,Sub,_,Subs,Value) :-
hf_branch_aromatic_pos(Sub,Subs,NumReg,NmN21BReg,NmNxt21BrgInner,
 NmNxt21BrgOuter,NmNxt22Brgs),
hf_ba_reg(aromatic,Val1),
hf_ba_breg(aromatic,Val2),
hf_ba_inn(aromatic,Val3),
hf_ba_out(aromatic,Val4),
hf_ba_mid(aromatic,Val5),
Part is NumReg*Val1 + NmN21BReg*Val2 + NmNxt21BrgInner*Val3 +
NmNxt21BrgOuter*Val4 + NmNxt22Brgs*Val5,
hf_data(connectivity,Sub,Part),

hf_get_sub_size(Sub,Subs,Sizes),
%% get_sub_size(Sub,Subs,Sizes),
sumlist(Sizes,Size),
length(Subs,Len),
ESize is Size - Len * 0.05,
maximum([0,ESize],EffectiveSize),
hf_sa(aromatic,Hfs),
Steric is EffectiveSize*Hfs,
        hf_data(steric,Sub,Steric),

hf_branch_ortho(Subs,Sub,Ortho),
%% hf_data(connectivity,Sub,Ortho),
hf_data(steric,Sub,Ortho),

hf_perri(Subs,Sub,0,Perri),
hf_data(steric,Sub,Perri),
Value is Part + Steric + Ortho + Perri.
hf_branch(oh,Sub,_,[],Value):-
Connectivity is 0,
        hf_data(connectivity,Sub,Connectivity),
        Value is Connectivity.
hf_branch(oh,Sub,_,[Subs],Value):-
reac_type(Subs,_,Type),
hf_connect_oh(Type,Val),

Connectivity is Val,
        hf_data(connectivity,Sub,Connectivity),

        Value is Connectivity.
hf_branch(co2h,Sub,_,[],Value):-
Connectivity is 0,
        hf_data(connectivity,Sub,Connectivity),
        Value is Connectivity.
hf_branch(co2h,Sub,_,[Subs],Value):-
reac_type(Subs,_,Type),
hf_connect_co2h(Type,Val),
Connectivity is Val,
        hf_data(connectivity,Sub,Connectivity),
        Value is Connectivity.
hf_branch(_SubType,_,_,_Subs,error).


/*******************************************************
Deals with the connectivity of Atoms.
-methyl - for methyl groups
when there are Len num of Subs,
get's correction based on polynomial +
correction for what each of the type those
subs are.
only necessary when there are more than 2
substituents b/c that's when this interaction
occurs
*******************************************************/
hf_branch_connectivity(methyl,Len,Sub,Subs,Connectivity):-
hf_branch_connectivity_aux(methyl,Len,Subs,0,C),

hf_branch_methyl(a,A),
hf_branch_methyl(b,B),
Connectivity is (A*Len*Len)+(B*Len) + C,
        hf_data(connectivity,Sub,Connectivity).
```

```
hf_branch_connectivity_aux(methyl,_,[],Out,Out).
hf_branch_connectivity_aux(methyl,Len,[Sub|Rest],Part,C):-
Len > 2,
reac_type(Sub,_,Name),

concatenate_atomlist_to_atom([Name-Len],Key),
hf_branch_methyl(Key,Val),

NewPart is Part + Val,
hf_branch_connectivity_aux(methyl,Len,Rest,NewPart,C).
hf_branch_connectivity_aux(methyl,Len,[_|Rest],Part,C):-
hf_branch_connectivity_aux(methyl,Len,Rest,Part,C).


/*******************************************************
gets a correction for acetylenic being endo in
a ring.
*******************************************************/
hf_endo_ace(Sub,Endo):-
reactophore_fact(Sub,_,[A,B],_,_),
findall(_X,(ring(_,R),
   memberchk(A,R),
   memberchk(B,R)),[_|_]),
hf_branch_acetylenic(endo,Endo),
!.
hf_endo_ace(_,0).

/*******************************************************
gets the number of subs that are in each of the 5 possible
positions in an aromatic hydrocarbon:
  regular location NOT next to a bridge
  regular location next to a bridge if cmpd is NOT bent
  inner location next to a bridge if bent
  outer location next to a bridge if bent
  middle location between 2 bridges

Sub is Sub sitting on....Subs are the substituents off that Sub

first gets the Atoms of the all the Subs in a list
--maybe should seperate to lists of atoms for each sub
then gets all the bridge_atoms in the Sub
next get All the Atoms in the Sub

then find the locations:
```

```
after the locations are found find out how much each location
contributes to the total Hf, b/c some branches might
be in more than 1 location: eg middle location and inner
figure out how much each should contribute by getting total num
of subs, and total num of locations,
the actual contribution for each location is:
the location/total num locations * total subs
*****************************************************/

hf_branch_aromatic_pos(_,[],0,0,0,0,0).
hf_branch_aromatic_pos(Sub,Subs,NumReg,NN21BReg,
NumNxt21BrgInner,NumNxt21BrgOuter,NumNxt22Brgs):-
hf_branch_aromatic_pos_aux(Subs,[],Atoms),
bridge_atoms(Sub,BridgeAs),
reactophore_fact(Sub,_,AA,_,_),
hf_branch_aromatic_pos_aux2a(Atoms,AA,BridgeAs,_,0,NumNxt22Brgs1),
hf_branch_aromatic_pos_aux2b(Atoms,AA,BridgeAs,_,0,NumNxt21BrgOuter1),
hf_branch_aromatic_pos_aux2c(Atoms,AA,BridgeAs,_,0,NumNxt21BrgInner1),
hf_branch_aromatic_pos_aux2d(Atoms,AA,BridgeAs,_,0,NN21BReg1),
hf_branch_aromatic_pos_aux2e(Atoms,AA,BridgeAs,_,0,NumReg1),

length(Subs,NumSubs),
Total is NumReg1 + NN21BReg1 + NumNxt21BrgInner1
 + NumNxt21BrgOuter1 + NumNxt22Brgs1,

NumReg is (NumReg1/Total)*NumSubs,
NN21BReg is (NN21BReg1/Total)*NumSubs,
NumNxt21BrgInner is (NumNxt21BrgInner1/Total)*NumSubs,
NumNxt21BrgOuter is (NumNxt21BrgOuter1/Total)*NumSubs,
NumNxt22Brgs is (NumNxt22Brgs1/Total)*NumSubs.

/*****************************************************
NumNxt22Brgs1 is Number of Subs Next To 2 Bridges
*****************************************************/
hf_branch_aromatic_pos_aux2a(_,_,[],_,_,0).
hf_branch_aromatic_pos_aux2a([],_,_,_,Part,Part).
hf_branch_aromatic_pos_aux2a([H|T],AA,BridgeAs,Subs,Part,NumNxt22Brgs1):-
findall(X,(member(X,H),
   bonded(X,_,Z),
     memberchk(Z,AA),
   bonded(Z,_,Y),
   Y \= X,
   memberchk(Y,BridgeAs),
   bonded(Z,_,YY),
   YY \= X,
   YY \= Y,
   memberchk(YY,BridgeAs)),Next22Bridge1),
delete_duplicates(Next22Bridge1,Next22Bridge),
length(Next22Bridge,NumN22B),
NewPart is Part + NumN22B,
hf_branch_aromatic_pos_aux2a(T,AA,BridgeAs,Subs,NewPart,NumNxt22Brgs1).

/*****************************************************
Outer is NumberofSubsNextToBridge on the Outer side of a curve
*****************************************************/
hf_branch_aromatic_pos_aux2b(_,_,[],_,_,0).
hf_branch_aromatic_pos_aux2b([],_,_,_,Part,Part).
hf_branch_aromatic_pos_aux2b([H|T],AA,BridgeAs,Subs,Part,Outer):-
findall(X,(member(X,H),
   bonded(X,_,Z),
     memberchk(Z,AA),
   bonded(Z,_,Y),
   Y \= X,
   memberchk(Y,BridgeAs),
   bonded(Z,_,YY),
   YY \= X,
   YY \= Y,
   \+ memberchk(YY,BridgeAs),
   bonded(YY,_,YY2),
   memberchk(YY2,AA),
   YY2 \= Z,
   memberchk(YY2,BridgeAs),
   bonded(YY2,_,YY3),
   YY3 \= YY,
   memberchk(YY3,BridgeAs),
   bonded(Y,_,Y2),
   memberchk(Y2,AA),
   Y2 \= Z,
   memberchk(Y2,BridgeAs),
   bonded(YY3,_,Y2)),ListOuter1),
delete_duplicates(ListOuter1,ListOuter),
length(ListOuter,NumOuter),
NewPart is Part + NumOuter,
hf_branch_aromatic_pos_aux2b(T,AA,BridgeAs,Subs,NewPart,Outer).

/*****************************************************
Inner is NumberofSubsNextToBridge on the Inner side of a curve
*****************************************************/
hf_branch_aromatic_pos_aux2c(_,_,[],_,_,0).
hf_branch_aromatic_pos_aux2c([],_,_,_,Part,Part).
```

```
hf_branch_aromatic_pos_aux2c([H|T],AA,BridgeAs,Subs,Part,Inner):-
findall(X,(member(X,H),
   bonded(X,_,Z),
     memberchk(Z,AA),
   bonded(Z,_,Y),
   Y \= X,
   memberchk(Y,BridgeAs),
   bonded(Y,_,Y2),
   memberchk(Y2,BridgeAs),
   bonded(Y,_,Y3),
   Y2 \= Y3,
   memberchk(Y3,BridgeAs),
   bonded(Y2,_,Y4),
   Y4 \= Y,
   memberchk(Y4,BridgeAs),
   bonded(Y2,_,Y8),
   Y8 \= Y,
   Y8 \= Y4,
   memberchk(Y8,AA),
   bonded(Y3,_,Y5),
   memberchk(Y5,AA),
   Y5 \= Y,
   bonded(Y4,_,Y6),
   Y6 \= Y2,
   memberchk(Y6,AA),
   bonded(Y5,_,Y6)),ListInner1),
delete_duplicates(ListInner1,ListInner),
length(ListInner,NumInner),
NewPart is Part + NumInner,
hf_branch_aromatic_pos_aux2c(T,AA,BridgeAs,Subs,NewPart,Inner).

/*********************************************************
NN2Brg is NumberNextToBridge
*********************************************************/
hf_branch_aromatic_pos_aux2d(_,_,[],_,_,0).
hf_branch_aromatic_pos_aux2d([],_,_,_,Part,Part).
hf_branch_aromatic_pos_aux2d([H|T],AA,BridgeAs,Subs,Part,NN2Brg):-
findall(X,(member(X,H),
   bonded(X,_,Z),
     memberchk(Z,AA),
   bonded(Z,_,Y),
   Y \= X,
   memberchk(Y,BridgeAs),
   bonded(Z,_,YY),
   YY \= X,
   YY \= Y,
   \+ memberchk(YY,BridgeAs),
   bonded(YY,_,YY2),
   memberchk(YY2,AA),
   YY2 \= Z,
   \+ memberchk(YY2,BridgeAs),
   bonded(Y,_,Y2),
   memberchk(Y2,AA),
   Y2 \= Z,
   \+ memberchk(Y2,BridgeAs),
   bonded(Y,_,Y3),
   memberchk(Y3,AA),
   Y3 \= Z,
   memberchk(Y3,BridgeAs)),ListN2Brg1),
delete_duplicates(ListN2Brg1,ListN2Brg),
length(ListN2Brg,NumN2Brg),
NewPart is Part + NumN2Brg,
hf_branch_aromatic_pos_aux2d(T,AA,BridgeAs,Subs,NewPart,NN2Brg).

/******************************************************
Regular Subs--ie not next to a bridge
******************************************************/
hf_branch_aromatic_pos_aux2e([],_,_,_,Part,Part).
hf_branch_aromatic_pos_aux2e([H|T],AA,BridgeAs,Subs,Part,Reg):-
findall(X,(member(X,H),
   bonded(X,_,Z),
     memberchk(Z,AA),
   bonded(Z,_,Y),
   Y \= X,
   \+ memberchk(Y,BridgeAs),
   bonded(Z,_,YY),
   YY \= X,
   YY \= Y,
   \+ memberchk(YY,BridgeAs)),ListReg1),
delete_duplicates(ListReg1,ListReg),
length(ListReg,NumReg),
NewPart is Part + NumReg,
hf_branch_aromatic_pos_aux2e(T,AA,BridgeAs,Subs,NewPart,Reg).

/******************************************************
hf_branch_aromatic_pos_aux:
gets the a list of the Atoms in all of the Subs
******************************************************/
hf_branch_aromatic_pos_aux([],Atoms,Atoms).
hf_branch_aromatic_pos_aux([H|T],List,Atoms) :-
```

```
reactophore_fact(H,_,Atom,_,_),
hf_branch_aromatic_pos_aux(T,[Atom|List],Atoms).


/***********************************************************************
For aromatic groups an ortho correction:
Subs are the groups off of Arom

hf_branch_ortho_aux gets the subs that are ortho to each other

hf_branch_ortho_aux2 takes all of the OrthoPairs in an aromatic ring
and depending on if the sub in question is
paired to only one other sub (outer) or
is between two other subs (inner)
      the routine follows the proper path to get the appropriate correction

***********************************************************************/
hf_branch_ortho(Subs,Arom,OrthoSum) :-
hf_branch_ortho_aux(Subs,Arom,OrthoPairs),
hf_branch_ortho_aux2(OrthoPairs,Arom,Ortho),
sumlist(Ortho,OrthoSum).

hf_branch_ortho_aux([],_,[]).
hf_branch_ortho_aux([H|T],Arom,[H-NewList|Rest]):-
ortho_subs(Arom,H,List),
hf_boa(List,H,NewList),
hf_branch_ortho_aux(T,Arom,Rest).

hf_boa([],_,[]).
hf_boa([H|T],Sub,[H|T]):-
reactophore_fact(Sub,_,[A],_,_),
findall(_X,(ring(_,R),
   length(R,LenR),
   LenR < 9,
   memberchk(A,R)),[]),
!.
hf_boa([H|T],Sub,[H|Rest]):-
reactophore_fact(Sub,_,[A],_,_),
findall(X,(ring(_,X),
%    length(R,LenR),
   length(X,LenR),
   LenR < 9,
   memberchk(A,X)),List),
reactophore_fact(H,_,[B],_,_),
findall(_Y,(member(AA,List),
   memberchk(B,AA)),[]),
!,
hf_boa(T,Sub,Rest).
hf_boa([_|T],Sub,Rest):-
hf_boa(T,Sub,Rest).

hf_branch_ortho_aux2([],_,[]).
hf_branch_ortho_aux2([_-[]|T],Arom,Ortho):-
hf_branch_ortho_aux2(T,Arom,Ortho).
hf_branch_ortho_aux2([H-[Sub]|T],Arom,[Part|Ortho]):-  %outer
hf_ortho(outer,A),
hf_branch_ortho_mult(outer,H,[Sub],Mult),
hf_get_sub_size(Arom,[H,Sub],Sizes),
sumlist(Sizes,Size),
ESize is Size - 2 * 0.05,
maximum([0,ESize], EffectiveSize),
hf_ortho(inner_outer_size,OS),
Part is A*Mult + EffectiveSize*OS,
hf_branch_ortho_aux2(T,Arom,Ortho).
hf_branch_ortho_aux2([H-[Sub1,Sub2]|T],Arom,[Part|Ortho]):- %inner
hf_ortho(inner,A),
hf_branch_ortho_mult(inner,H,[Sub1,Sub2],Mult),
hf_get_sub_size(Arom,[H,Sub1,Sub2],Sizes),
sumlist(Sizes,Size),
ESize is Size - 3 * 0.05,
maximum([0,ESize], EffectiveSize),
hf_ortho(inner_outer_size,IS),
Part is A*Mult + EffectiveSize*IS,
hf_branch_ortho_aux2(T,Arom,Ortho).

/**************************************************************************
gets the multiplier for different substiuents besides methyl
both are methyl - 1
both are cl - -5, etc
**************************************************************************/
hf_branch_ortho_mult(outer,H,[Sub],Mult):-
reac_type(H,_,TypeH),
reac_type(Sub,_,TypeS),
hf_branch_ortho_mult_aux(outer,H,TypeH,TypeS,_,Mult).
hf_branch_ortho_mult(outer,_,_,1).
hf_branch_ortho_mult(inner,H,[Sub1,Sub2],Mult):-
reac_type(H,_,TypeH),
reac_type(Sub1,_,Type1),
reac_type(Sub2,_,Type2),
hf_branch_ortho_mult_aux(inner,H,TypeH,Type1,Type2,Mult).
```

```
hf_branch_ortho_mult(inner,_,_,1).

%%%%
hf_branch_ortho_mult_aux(outer,_,Type,Type,_,Mult):-
hf_ortho(outer-Type,Mult).
hf_branch_ortho_mult_aux(outer,_,TypeH,TypeS,_,Mult):-
TypeH \= TypeS,
sort([TypeH,TypeS],List),
concatenate_atomlist_to_atom(List,Key),
hf_ortho(outer-Key,Mult).
hf_branch_ortho_mult_aux(outer,_,_,_,_,1).

%%%%
hf_branch_ortho_mult_aux(inner,_,Type,Type,Type,Mult):-
hf_ortho(inner-Type,Mult).
hf_branch_ortho_mult_aux(inner,H,TypeH,Type1,Type2,Mult):-
hf_branch_ortho_mult_aux_aux(inner,H,TypeH,Type1,M1),
hf_branch_ortho_mult_aux_aux(inner,H,TypeH,Type2,M2),
Mult is M1 + M2.

/***
sort([TypeH,Type1,Type2],List),
concatenate_atomlist_to_atom(List,Key),
hf_boma(Key,H,MultMult),
hf_ortho(inner-Key,X),
Mult is MultMult*X.
****/

hf_branch_ortho_mult_aux(inner,_,_,_,_,1).


hf_branch_ortho_mult_aux_aux(inner,_,Type,Type,M1):-
hf_ortho(inner-Type,M1).
hf_branch_ortho_mult_aux_aux(inner,H,methyl,f,M1):-
reactophore_fact(H,_,_,_,Atoms),
findall(F,(member([A],Atoms),
   reactophore_fact(F,_,At1,_,_),
   memberchk(A,At1),
   reac_type(F,_,f)),[_,_,_]),
hf_ortho(inner-methylf3,M1).
hf_branch_ortho_mult_aux_aux(inner,_,Type1,Type2,M1):-
sort([Type1,Type2],List),
concatenate_atomlist_to_atom(List,Key),
hf_ortho(inner-Key,M1).
```

```
hf_boma(fmethyl,H,MultMult):-
findall(F,(pair(H,F),
   reac_type(F,_,f)),LF),
length(LF,NumF),
NumF =3,
hf_ortho(multf,MultMult).
hf_boma(_,_,1).


/************************************************************************
correction for Perri pairs of atoms
************************************************************************/
hf_perri([],_,Perri,Perri).
hf_perri([H|T],Sub,Part,Perri) :-
reactophore_fact(H,_,[A],_,_),
findall(R,(ring(_,R),
           length(R,LenR),
   LenR < 9,
   memberchk(A,R)),[]),
findall(X,perri_pair(H,X), List),
List \= [],
hf_get_sub_size(Sub, List, Sizes),
%% get_sub_size(Sub, List, Sizes),
sumlist(Sizes, Size),
ESize is Size,
hf_perri_param(aromatic, Hfs),
NewPart is Part + Hfs * ESize,
!,
hf_perri(T,Sub,NewPart,Perri).
hf_perri([_|T],Sub,Part,Perri):-
hf_perri(T,Sub,Part,Perri).
hf_perri(_,_,_,0).


/************************************************************************
Correction for groups that are next to methyl groups, that are not in a ring
The groups being examined have to have 3 substiuents and 1 bond to the other
group for a total of 4 things off it.

        Basically, another steric correction
************************************************************************/
hf_near_methyl(RingAtoms,Near) :-
%% findall(Q,ring(_,Q),ListQ),%% Changed 2/25/04
%% flatten(ListQ,RingAtoms),
findall(X,(substituent_type(X,AA,_),
   \+ memberchk(AA,RingAtoms),
   atom_specs(_,AA,_,_,[A1,A2,A3,A4]),
   hf_near_methyl_aux([A1,A2,A3,A4],[S1,S2,S3,S4]),
   hf_get_sub_size(X,[S1,S2,S3,S4],Sizes),
   findall(S,(member(S,Sizes),
      S > 0.05),[_,_,_,_])
                 ),List4),
findall(Y,(member(Y,List4),
   pair(Y,Z),
   Z \= Y,
   memberchk(Z,List4)),List),
length(List,Len),
hf_near_methylparam(4,V),
Near is V*Len.

hf_near_methyl_aux([],[]).
hf_near_methyl_aux([H|T],[S1|List]):-
reactophore_fact(S1,_,AAA,_,_),
memberchk(H,AAA),
hf_near_methyl_aux(T,List).

/**********************************************************************
This actually returns corrections for the steric interaction
of groups that are connected to the db.
A   B
  =
C   D
hf_cis_1 handles (A&C) and (B&D) interactions
hf_cis_2 handles (A&B) and (C&D) interactions

1 is for chains,
2 is for db's in rings,
3 is in case everything else fails
**********************************************************************/
hf_cis_trans(Sub,Type,Value) :-
reactophore_fact(Sub,e,[A,B],_,[AA,BB]),%A,B-atoms that makeup db
findall(X,(ring(_,X),%%AA and BB are the atoms connected to A&B
   intersects([A,B],X,[_,_])),[]),%make sure not in a ring--endo
findall(Y,(ring(_,Y),
   intersects([A,B],Y,[_])),[]),%make sure not in a ring--exo
!,
hf_cis(Type,Type,A,B,AA,BB,Sub,Value).
hf_cis_trans(Sub,Type,Value) :-
reactophore_fact(Sub,e,[A,B],_,[_AA,_BB]),
findall(Y-X,(ring(_,X),%finds rings w/ endo ethylenics
     length(X,Y),
     intersects([A,B],X,[_,_])),EndoRing1),
(EndoRing1 == [] -> EndoRing = []
  ; keysort(EndoRing1,SortEndoRing),
    filter_endoexo(SortEndoRing,Sub,EndoRing)
),
findall(Y-X,(ring(_,X),%finds rings w/ exo ethylenics
     length(X,Y),%off of the A-Sub
     intersects([A],X,[_])),ExoRing1A),
findall(Y-X,(ring(_,X),%finds rings w/ exo ethylenics
     length(X,Y),%off of the B-Sub
     intersects([B],X,[_])),ExoRing1B),
(ExoRing1A == [] -> ExoRingA =[]
   ; keysort(ExoRing1A,SortExoRingA),
          filter_endoexo_top(SortExoRingA,Sub,ExoRingA)
),
(ExoRing1B == [] -> ExoRingB =[]
   ; keysort(ExoRing1B,SortExoRingB),
          filter_endoexo_top(SortExoRingB,Sub,ExoRingB)
),
        append(ExoRingA,ExoRingB,ExoRing),
filter_endoexo2(ExoRing,EndoRing,NewExoRing),

hf_ct_ring(Sub,Type,EndoRing,NewExoRing,Value).
%routine where all the real work starts
hf_cis_trans(_,_,0).

/**********************************************************************
This routine throws away the ExoRings where the db is also in
EndoRing
**********************************************************************/
filter_endoexo2(_,EndoRing,NewExoRing):-
EndoRing \= [],
NewExoRing =[].
filter_endoexo2(ExoRing,[],ExoRing).


/**********************************************************************
The _top call is for those rings with a db on the bridge of noraborane
aka two 5membered rings joined w/ a bridge.

This routine is to get the most strained rings in a molecule,
it throws away the rings where the db appears twice, but the
correction should only get picked up once.
```

```
***********************************************************************/
filter_endoexo_top(SortedRings,Sub,Out):-
findall(Len,member(Len-X,SortedRings),ListLen),
(delete_duplicates(ListLen,[5]) -> findall(X,member(Len-X,SortedRings),Out)
    ; filter_endoexo(SortedRings,Sub,Out)
).

filter_endoexo([_-H|T],Sub,[H|Out]):-%List are the atoms hooked to Sub that
findall(X,(pair(Sub,_,X,_),%are in the ring (H)
   memberchk(X,H)),List),
filter_endoexo_aux(T,Sub,List,Out).

filter_endoexo_aux([],_,_,[]).
filter_endoexo_aux([_-H|T],Sub,List,Out):-
%throws away rings that are part of List in at
reactophore_fact(Sub,e,[C1,C2],_,_),%least 2 places
difference(H,[C1,C2],RestAtoms),
intersects(RestAtoms,List,[_|_]),
!,
filter_endoexo_aux(T,Sub,List,Out).
filter_endoexo_aux([_-H|T],Sub,List,[H|Out]):-%otherwise keep the ring
filter_endoexo_aux(T,Sub,List,Out).


/***********************************************************************
gets the correction for rings that have Endo and Exo db's in them
***********************************************************************/
hf_ct_ring(Sub,Type,Endo,Exo,Value):-
hf_ct_ring_endo(Endo,Endo,Sub,Type,0,ValEndo),
hf_ct_ring_exo(Exo,Exo,Sub,Type,0,ValExo),
hf_halo(ethylenic,Sub,Halo),

Value is ValEndo+ValExo + Halo.

/***********************************************************************
Exo db rings
Gets a correction for exo's based on size of ring that db is exo to
***********************************************************************/
hf_ct_ring_exo([],_,_,_,Part,Part).
hf_ct_ring_exo([Ring|T],AllGood,Sub,Type,Part,ValExo):-
length(Ring,RingSize),
hf_ring_exo_aux(Sub,Type,Ring,RingSize,Val),%gets the basic exo correction
hf_data(basic_ring_exo,Sub,Val),

reactophore_fact(Sub,e,[_C1,_C2],_,[[A1,A2],[B1,B2]]),
flatten([[A1,A2],[B1,B2]],FlatSubs),
findall(X,(member([X],[A1,A2]),
   memberchk(X,Ring)),RA),
(length(RA,0);length(RA,2)),
!,
findall(X,(member([X],[B1,B2]),
   memberchk(X,Ring)),RB),
(length(RB,0);length(RB,2)),
!,
append(RA,RB,RAtoms),
difference(FlatSubs,RAtoms,Others),
findall(Y,(ring(_,Y),%gets the ring on the other side of the db
   \+ Y = Ring,%if it's empty call ..._aux
   memberchk(Y,AllGood),%else, call ..._aux2
   member(Other1,Others),
   memberchk(Other1,Y),
   member(Other2,Others),
   \+ Other2 = Other1,
   memberchk(Other2,Y)),OtherRingX),
delete_duplicates(OtherRingX,OtherRing),
(OtherRing = [] -> hf_ct_ring_exo_aux_top(Others,Sub,Type,
[[A1,A2],[B1,B2]],RingSize,Val2)
         ; hf_ct_ring_exo_aux2(OtherRing,Others,Sub,Type,RingSize,0,Val2)
),

hf_ct_ring_exo_aux3(RingSize,Ring,Sub,Val3),
hf_data(steric1,Sub,Val3),

NewPart is Part + Val + Val2 + Val3,
hf_ct_ring_exo(T,AllGood,Sub,Type,NewPart,ValExo).
hf_ct_ring_exo(_,_,_,_,_,_):-
write('ERROR ERROR ERROR'),
abort.


/************************************************************************
Takes into account the strain additional exos to a ring
place on the ring.

ortho interaction - for extra exos in the ortho position

ring deform - for exos in pos other than ortho, should only affect 4 ring

***********************************************************************/
hf_ct_ring_exo_aux3(RingSize,Ring,Sub,Val3):-
```

```
%%%% ortho interaction
findall(X,(pair(Sub,X),
   reactophore_fact(X,e,[A,B],_,_),
   intersects([A,B],Ring,[_])),OrthoBondedExos),
hf_ct_ring_exo_aux3_aux1(OrthoBondedExos,RingSize,Ortho),

%%%% ring deform
findall(X,(reactophore_fact(X,e,[A,B],_,_),
   X \= Sub,
   intersects([A,B],Ring,[_]),
   \+ pair(Sub,X)),OtherExos),
hf_ct_ring_exo_aux3_aux2(OtherExos,RingSize,RingDeform),

Val3 is Ortho + RingDeform.

/***************************************************************************
For exos in positions on ring besides 1,2 gets the deformation
correction....only noticible in 4ring
***************************************************************************/
hf_ct_ring_exo_aux3_aux2([],_,0).
hf_ct_ring_exo_aux3_aux2([_|_],4,Val):-
hf_ring_exo(ringdeform,Val).
hf_ct_ring_exo_aux3_aux2([_|_],_,0).


/***************************************************************************
For extra exos in the ortho position, all have a steric interacion

For size 3 rings, there is no steric interaction, but something
lowers the value by ~21/ortho, could be extra resonance b/c
the dbs are nearly linear? or just really crappy data? based on
CAS 3227905

Correction for exos in the 1,2 positions (~6kJ)
For size 4 rings, this should get less of correction (half?)

Also corrects for the deformation of the ring due to ortho...none in set
right now but could be trained:
in 6 makes ring like a chair,
in 7 should be ok,
in a 8 deformed,
and >=9 should be ok

this is for single rings, i don't know what's going
```

```
to happen if the other exo is in another ring
*************************************************************************/
hf_ct_ring_exo_aux3_aux1([],_,0).
hf_ct_ring_exo_aux3_aux1(Exos,3,Val):-
length(Exos,LExos),
hf_ring_exo(ortho-3,V),
Val is LExos * V.
hf_ct_ring_exo_aux3_aux1([_|_],RingSize,Ortho):-
(RingSize > 4 -> Mult is 1
       ; (RingSize =:= 4 -> Mult is 0.5
  ; Mult is 0
 )
),
hf_ring_exo(ortho,OVal),
OrthoBasic is OVal * Mult,

concatenate_atomlist_to_atom([ortho-RingSize],Key),
%none in set so set equal to 0
hf_ring_exo(Key,OrthoDeform),

Ortho is OrthoBasic + OrthoDeform.

/*************************************************************************
Gets the basic exo correction
*************************************************************************/
hf_ring_exo_aux(_Sub,ethylenic,_Ring,RingSize,Val):-
(RingSize > 6 -> hf_ring_exo(other,Val1)
       ; hf_ring_exo(RingSize,Val1)
),
/**
(RingSize < 5 -> findall(X,(pair(Sub,X),
        reactophore_fact(X,e,[A,B],_,_),
        intersects([A,B],Ring,[_])),OtherBondedExos),
 length(OtherBondedExos,Len),
 (Len =:= 0 -> Mult is 1
     ; Mult is 0.75
 )
       ; Mult is 1
),
Val is Val1*Mult.
***/

Val is Val1.
hf_ring_exo_aux(_Sub,oethylenic,_Ring,RingSize,Val):-
(hf_ring_exo_oethylenic(RingSize,Basic) -> true
%if in db get val, else use 12's value
 ; hf_ring_exo_oethylenic(12,Basic)
),
%%% hf_ring_exo_sigma(oethylenic,Ring,Sigma),
%%not using b/c can't see it's effects easily
%%% Val is Basic + Sigma.
Val is Basic.

hf_ring_exo_aux(_Sub,nethylenic,_Ring,_RingSize,Val):-
Val is 0.
hf_ring_exo_aux(_Sub,sethylenic,_Ring,_RingSize,Val):-
Val is 0.

/***************************************************************************
I don't think this correction is necessary yet - it's not easily
observable - therefore I'm not using it just yet!

This get's a "sigma" correction for groups off of the carbon next
to the "=", this stablizes the engery by ~6kJ/methyl.

Perhaps this should be put higher up, so that it is picked up in
every "ethy" type group, now this is subsumed by the steric correction
- if it were seperate maybe would show up in that too!!!
***************************************************************************/
hf_ring_exo_sigma(oethylenic,Ring,Sigma):-
n_o_s_type(Sub,oethylenic),
reactophore_fact(Sub,_,Name,_,Subs),
flatten(Subs,FSubs),
findall(Sub1,(member(Sub1,FSubs),
      memberchk(Sub1,Ring)),NewRSubs),

hf_res_branches(NewRSubs,Ring,Name,0,Branches),
hf_exo_sigma(pp,Sig),
Sigma is Branches*Sig.

hf_res_branches([],_,_,Out,Out).

hf_res_branches([H|T],Ring,Name,Part,Branches):-
reactophore_fact(_,_,[H],_,Subs),
flatten(Subs,FSubs),
findall(APart,(member(APart,FSubs),
       \+ (memberchk(APart,Ring))),ListB),
length(ListB,LenB),
hf_res_branch_mult(H,Name,Ring,Mult),%this is a temporary stub!!!!
Val is LenB * Mult,
NewPart is Part + Val,
hf_res_branches(T,Ring,Name,NewPart,Branches).

hf_res_branch_mult(H,Name,Ring,Mult):-
bonded(H,_,Sub),
memberchk(Sub,Ring),
\+ memberchk(Sub,Name),
reac_type(Sub,Type),
(Type = ethylenic -> Mult is 0.5
           ; Mult is 1
).


/************************************************************************
modified get_sub_size:
Sub is the substituent you are on
List is the list of subs coming off of Sub
Sizes is the list of sizes of the subs

look in steric.pro for doing(hf) for ring thing, may need to change

doesn't return anything that is larger than 2-t-butyls
************************************************************************/
hf_get_sub_size(Sub,List,Sizes):-
get_sub_size(Sub,List,Sizes0),
hf_get_sub_size_aux(Sizes0,Sizes).

hf_get_sub_size_aux([],[]).
hf_get_sub_size_aux([H|T],[Max|Sizes]):-
% minimum([H,0.367],Max),%1-t-butyl * 2
minimum([H,0.531],Max),%2-t-butyls
hf_get_sub_size_aux(T,Sizes).

/************************************************************************
gets the correction for exo-rings that have branches coming off of
the Subs on the other side of the double bond.
the 1 branch is:
similar to hf_cis_2....uses some of the same database values
the 2 branch is:
similar to hf_cis_1....uses some of the same database values
    and similar to hf_cis_2...also uses same database vals.

************************************************************************/
hf_ct_ring_exo_aux_top([],_,_,_,_,0).
hf_ct_ring_exo_aux_top(Others,Sub,Type,[[A1,A2],[B1,B2]],RingSize,Val2):-
```

```
findall([X1,Y1],(member(X1,Others),
 (([X1] = A1, [Y1] = B2);
  ([X1] = A2, [Y1] = B1);
  ([X1] = B1, [Y1] = A2);
  ([X1] = B2, [Y1] = A1)
 )),OtherPairs),
hf_ct_ring_exo_aux(Others,Sub,Type,OtherPairs,RingSize,Val2).


hf_ct_ring_exo_aux([_],Sub,_,_,_,Value):-%when there is a n_o_s
neth_oeth_seth(Sub),
Value is 0,
hf_data(steric1,Sub,Value).
hf_ct_ring_exo_aux([A1],Sub,_,[[A1,B1]],_RingSize,Value):-
%when there is only 1 branch
reactophore_fact(SubA1,_,AL,_,_),
memberchk(A1,AL),
reactophore_fact(SubB1,_,BL,_,_),
memberchk(B1,BL),

hf_get_sub_size(Sub,[SubA1,SubB1],SizesAB),
hf_cis_2_aux(SizesAB,[AL,BL],Value),
hf_data(steric1,Sub,Value).

hf_ct_ring_exo_aux([_A1,_A2],_Sub,nethylenic,_,_RingSize,Value):-

%when there is a neth w/ a branch
%going to be similar to things w/ one thing coming off

Value is 0.

hf_ct_ring_exo_aux([A1,A2],Sub,_Type,[[A1,B1],[A2,B2]],_RingSize,Value):-
%when there are 2 branches
%%% cis_2 part
hf_ct_ring_exo_aux([A1],Sub,_,[[A1,B1]],_,Val21),
hf_ct_ring_exo_aux([A2],Sub,_,[[A2,B2]],_,Val22),

%%% cis_1 part
        pair(Sub,_,A1,SubA1),
        pair(Sub,_,A2,SubA2),
        hf_get_sub_size(Sub,[SubA1,SubA2],SizesA),
hf_cis_1(SizesA,[SubA1,SubA2],Sub,Val1),

        Value is Val1 + Val21 + Val22.
```

```
/*************************************************************************
doesn't need a correction right now b/c counts the 2 rings up in
hf_ct_ring_exo
*************************************************************************/
 hf_ct_ring_exo_aux2(_,_,_,_,_,0,0).


/*************************************************************************
Gets a basic correction for size of the db_ring on,
Others are the branches (atoms) off of the ring
Calls routines that gets the other correction for interaction between
branches, other rings, etc, depends on whether or not
OtherRingX is empty or not.
*************************************************************************/
hf_ct_ring_endo([],_,_,_,Part,Part).
hf_ct_ring_endo([Ring|T],AllGood,Sub,Type,Part,ValEndo):-
length(Ring,RingSize),
get_hf_ring_db(RingSize,Sub,Val1),%basic correction
hf_data(basic_ring_endo,Sub,Val1),

reactophore_fact(Sub,e,[C1,C2],_,[[A1,A2],[B1,B2]]),
flatten([[A1,A2],[B1,B2]],FlatSubs),
findall(X,(member(X,FlatSubs),
   memberchk(X,Ring)),[R1,R2]),
difference(FlatSubs,[R1,R2],Others),
findall(Y,(ring(_,Y),%gets the ring on the other side of the db
   \+ Y = Ring,%if it's empty call ..._aux
   memberchk(Y,AllGood),%else, call ..._aux2
   memberchk(C1,Y),
   memberchk(C2,Y),
   (Others = [A,A] -> memberchk(A,Y)
    ; member(Other1,Others),
          memberchk(Other1,Y),
          member(Other2,Others),
      \+ Other2 = Other1,
      memberchk(Other2,Y)
   )),OtherRingX),

delete_duplicates(OtherRingX,OtherRing),
(OtherRing=[] -> hf_ct_ring_endo_aux_top(Others,Sub,Type,[[A1,A2],[B1,B2]],
 Ring,RingSize,Val2)
       ; hf_ct_ring_endo_aux2(OtherRing,Others,Sub,Type,
        Ring,RingSize,0,Val2)
),
NewPart is Part + Val1 + Val2,
hf_ct_ring_endo(T,AllGood,Sub,Type,NewPart,ValEndo).

/**********************************************************************
gets the correction for rings that have branches coming off of
the Subs on the other side of the double bond.
the 1 branch is:
similar to hf_cis_1....uses some of the same database values
the 2 branch is:
similar to hf_cis_1....uses some of the same database values
    and similar to hf_cis_2...also uses same database vals.


!!!NEEDS TO BE REWRITTEN SO MORE COMPACT AND RESEMBLES hf_cis more closely
**********************************************************************/
hf_ct_ring_endo_aux_top([],_,_,_,_,_,0).
hf_ct_ring_endo_aux_top(Others,Sub,Type,[[A1,A2],[B1,B2]],
Ring,RingSize,Val2):-
findall([X1,Y1],(member(X1,Others),
 (([X1] = A1, [Y1] = A2);
  ([X1] = A2, [Y1] = A1);
  ([X1] = B1, [Y1] = B2);
  ([X1] = B2, [Y1] = B1)
 )),OtherPairs),
hf_ct_ring_endo_aux(Others,Sub,Type,OtherPairs,Ring,RingSize,Val2).

hf_ct_ring_endo_aux([A1],Sub,Type,[[A1,B1]],Ring,RingSize,Value):-
%when there is only 1 branch
reactophore_fact(SubA1,_,AL,_,_),
memberchk(A1,AL),
reactophore_fact(SubB1,_,BL,_,_),
memberchk(B1,BL),

findall(X,(ring(_,X),%%what the hell is this doing?
   X \= Ring,
   memberchk(A1,X),
   memberchk(B1,X),
   findall(Y,bonded(B1,_,Y),[_,_])),List),
(List \= [] -> List = [_],
       Value is 0
      ; hf_cis_1_res([[A1],[B1]],0,Resonance),
                 hf_data(resonance1,Sub,Resonance),

       hf_get_sub_size(Sub,[SubA1,SubB1],[SizeA1,SizeB1]),
%size of the sub and ringsub
       hf_steric_ring(RingSize,Mult),
       Size is SizeA1 + SizeB1*Mult,
       ESize is Size - (2 * 0.05),
                maximum([0,ESize], EffectiveSize),
                n_o_s_type(Sub,Type),
                hf_cis_1_aux(Type,C),
               Steric is C*exp(EffectiveSize),
              hf_data(steric1,Sub,Steric),

       Value is Resonance + Steric
).
hf_ct_ring_endo_aux([_],_,_,_,_,_,0):-
write('ERROR ERROR ERROR'),
abort.
hf_ct_ring_endo_aux([A1,A2],Sub,Type,[[A1,B1],[A2,B2]],
Ring,RingSize,Value):-
%when there are 2 branches
%%% cis_1 part
hf_ct_ring_endo_aux([A1],Sub,Type,[[A1,B1]],Ring,RingSize,Val11),
hf_ct_ring_endo_aux([A2],Sub,Type,[[A2,B2]],Ring,RingSize,Val12),
%%% cis_2 part
        pair(Sub,_,A1,SubA1),
        pair(Sub,_,A2,SubA2),
        hf_get_sub_size(Sub,[SubA1,SubA2],SizesA),
hf_cis_2_aux(SizesA,[A1,A2],Val2),
hf_data(steric2,Sub,Val2),
Value is Val11 + Val12 + Val2.

/*************************************************************************
gets correction for rings that are joined to another ring across
a double-bond (eg ring||ring)

Others are the other atoms that were not part of the original ring,
can either have 1 or 2 things in them.

_aux2_aux gets a correction like hf_cis_1
*************************************************************************/
hf_ct_ring_endo_aux2([],_,_,_,_,_,Part,Part).
hf_ct_ring_endo_aux2([H|T],Others,Sub,_Type,_,RingSize,Part,Value):-
intersects(H,Others,Iatoms),
length(H,OtherRingSize),
hf_ct_ring_endo_aux2_aux(H,Iatoms,Sub,OtherRingSize,RingSize,Val),
NewPart is Part + Val,
hf_ct_ring_endo_aux2(T,Others,Sub,_Type,_,RingSize,NewPart,Value).

/*************************************************************************
```

```
...aux2_aux gets correction like hf_cis_1,
no need for a hf_cis_2 type correction b/c ring size is
picked up in hf_ct_ring_endo

not real sure about ...aux2_aux b/c don't understand Iatoms, if it's
one, then won't mess up when have a    \ ring1 ?
  ring2  ===

 b/c ring2 is exo and ring1 is endo

1st rule handles 3 and 4 rings joined across db.  No cis_1 interaction
    just a correction that raises the dHf by ~21
2nd rule is for a ring > 4 hooked to a 3 or 4 and
    for a 3 or 4 ring hooked to a ring > 4
    doesn't need a correction...well, that's what works
3rd rule is for those rings > 4
    should just get a correction like hf_cis_1...no additional strain
*************************************************************************/
hf_ct_ring_endo_aux2_aux(_H,_Iatoms,Sub,OtherRingSize,RingSize,Val):-
OtherRingSize =< 4,
RingSize =< 4,
%% reactophore_fact(Sub,_,As,_,_),
%% difference(H,As,Atoms),%Atoms not in the ring
%% length(Atoms,Len),
hf_ring_db(twoing,V),
Val is V,
hf_data(steric,Sub,Val).
hf_ct_ring_endo_aux2_aux(_H,_Iatoms,Sub,OtherRingSize,RingSize,Val):-
(OtherRingSize =< 4, RingSize > 4; OtherRingSize > 4, RingSize =< 4),
Val is 0,
hf_data(steric,Sub,Val).
hf_ct_ring_endo_aux2_aux(_H,Iatoms,Sub,OtherRingSize,RingSize,Val):-
OtherRingSize > 4,
RingSize > 4,
length(Iatoms,LenIatoms),
hf_ring_element_size(OtherRingSize,SizeORS1),
maximum([0,SizeORS1],SizeORS),
hf_ring_element_size(RingSize,SizeRS1),
maximum([0,SizeRS1],SizeRS),

/***
%%% I guess this used to be used but is not anywhere else in this section
Resonance is -7.1 * 2,
hf_data(resonance1,Sub,Resonance),
****/
```

```
ESizeORS is (SizeORS + SizeRS)-(LenIatoms * (2 * 0.05)),
maximum([0,ESizeORS],EffectiveSize),

        hf_steric_ethylenic(pp,C),

hf_ct_ring_endo_aux2_aux_halo(Iatoms,0,Halo),

Val is Halo + EffectiveSize * (LenIatoms * C),

hf_data(steric,Sub,Val).
hf_ct_ring_endo_aux2_aux(_H,_,Sub,_,_,Val):-
%catches anything that slips through
Val is 0,
hf_data(steric,Sub,Val).

/***************************
This is a fucking kludge, but maybe it'll work

supposed to see what the Iatoms are hooked to -
and get correction based on this
right now only correction is for bonded to two f's b/c stablizes by -32 each
***************************/
hf_ct_ring_endo_aux2_aux_halo([],Val,Val).
hf_ct_ring_endo_aux2_aux_halo([H|T],Part,Val):-
hf_halo_aux2_ethy1(_,H,Halos),
delete_duplicates(Halos,[f]),
length(Halos,Len),
hf_halo(ringendo,V),
NewPart is Part + V*Len,
hf_ct_ring_endo_aux2_aux_halo(T,NewPart,Val).
hf_ct_ring_endo_aux2_aux_halo([_|T],Part,Val):-
hf_ct_ring_endo_aux2_aux_halo(T,Part,Val).


/***************************************************************************
given a ring element get a size for it
***************************************************************************/
hf_ring_element_size(RingSize,Size):-
RingSize < 8,
hf_relsz(RingSize,Size).
hf_ring_element_size(RingSize,Size):-
RingSize >= 8,
Size is 0.05.
hf_ring_element_size(_,0).
```

```
/***********************************************************************
gets the correction for db's w/ branches coming off and interacting
based only on size and location
A&B--atoms that make up db
AA&BB--atoms that hooked to A&B respectively
Sub is the db you're on

_cis_aux2 gets the size(s) of the atom(s) off of the C
calls hf_get_sub_size, which caps the size at 0.189
_cis_1 deals w/ the interaction of atoms on the same C
_cis_2 deals w/ the strain of atoms on the same side of
the db--top and bottom
_connectivity gets the diff in ethy connected to nothing and
connected to something
_hbonding - gets the lowering of energy for hydrogen bonding
***********************************************************************/
hf_cis(ethylenic,Type,A,B,AA,BB,Sub,Value) :-
hf_cis_aux2(AA,A,Sub,SizesA),
hf_cis_aux2(BB,B,Sub,SizesB),
hf_cis_1(SizesA,AA,Sub,V1a),
hf_cis_1(SizesB,BB,Sub,V1b),
hf_cis_2(SizesA,AA,SizesB,BB,Sub,V2),
hf_cis_connectivity(Type,AA,BB,Sub,Connectivity),
hf_halo(ethylenic,Sub,Halo),
Value is V1a + V1b + V2 + Connectivity + Halo.
hf_cis(oethylenic,Type,A,B,AA,BB,Sub,Value) :-
hf_cis_aux2(AA,A,Sub,SizesA1),
atom_type(A,AType),
(AType = 'O' -> SizesA = [0,0]
      ; SizesA = SizesA1
),
hf_cis_aux2(BB,B,Sub,SizesB1),
atom_type(B,BType),
(BType = 'O' -> SizesB = [0,0]
      ; SizesB = SizesB1
),
hf_cis_1(SizesA,AA,Sub,V1a),
hf_cis_1(SizesB,BB,Sub,V1b),

hf_cis_connectivity(Type,AA,BB,Sub,Connectivity),

/***
(SizesA = [0,0] -> hf_cis_hbonding(Type,BB,B,0,HBonding)
         ; hf_cis_hbonding(Type,AA,A,0,HBonding)
),
hf_data(hbonding,Sub,HBonding),
Value is V1a + V1b + Connectivity + HBonding.
***/

Value is V1a + V1b + Connectivity.

hf_cis(nethylenic,Type,A,B,AA,BB,Sub,Value) :-
hf_cis_aux2(AA,A,Sub,SizesA),
hf_cis_aux2(BB,B,Sub,SizesB),
hf_cis_1(SizesA,AA,Sub,V1a),
hf_cis_1(SizesB,BB,Sub,V1b),
hf_cis_connectivity(Type,AA,BB,Sub,Connectivity),
hf_cis_hbonding(Type,AA,A,0,HBonding),
Value is V1a + V1b + Connectivity + HBonding.
hf_cis(sethylenic,Type,A,B,AA,BB,Sub,Value) :-
hf_cis_aux2(AA,A,Sub,SizesA),
hf_cis_aux2(BB,B,Sub,SizesB),
hf_cis_1(SizesA,AA,Sub,V1a),
hf_cis_1(SizesB,BB,Sub,V1b),
hf_cis_connectivity(Type,AA,BB,Sub,Connectivity),
hf_cis_hbonding(Type,AA,A,0,HBonding),
Value is V1a + V1b + Connectivity + HBonding.


/**********************************************************************
Determines if Sub is hooked to an arom or ethy and any branches
coming off 1-away have at least 1 H on it, if so H-bonding may
occur - lowering energy by ~15kJ
**********************************************************************/
hf_cis_hbonding(oethylenic,[],_,Out,Out).
hf_cis_hbonding(oethylenic,[[H]|T],Atom,In,HBonding):-
reactophore_fact(B,_,Atoms,_,_),
        memberchk(H,Atoms),
reac_type(B,_,Type),
memberchk(Type,[aromatic,ethylenic]),
!,
findall(Y,(bonded(H,_,X),
           memberchk(X,Atoms),
   bonded(X,_,Y),
           \+ memberchk(Y,Atoms)),ListY),
hf_cis_hbonding_aux(ListY,0,0,Part),
NewPart is In + Part,
hf_cis_hbonding(oethylenic,T,Atom,NewPart,HBonding).
hf_cis_hbonding(oethylenic,[_|T],Atom,In,HBonding):-
hf_cis_hbonding(oethylenic,T,Atom,In,HBonding).
hf_cis_hbonding(_,_,_,_,0).


hf_cis_hbonding_aux([],_,Out,Out).
hf_cis_hbonding_aux([H|T],Counter,In,Val):-
reactophore_fact(B,_,Atoms,_,Connect),
memberchk(H,Atoms),
reac_type(B,_,Type),
hf_cis_hbonding_aux_aux(Type,Connect,Counter,CounterOut,Part),
NewIn is In + Part,
hf_cis_hbonding_aux(T,CounterOut,NewIn,Val).

hf_cis_hbonding_aux([_|T],In,Part):-
hf_cis_hbonding_aux(T,In,Part).


hf_cis_hbonding_aux_aux(Type,Connect,Counter,NewCounter,Part):-
(Counter > 0 -> fail
      ;
memberchk(Type,[methyl,ethylenic]),
length(Connect,Len),
Len < 3,
hf_hbonding(pp,Part),
NewCounter = Counter + 1
).
hf_cis_hbonding_aux_aux(_,_,0,0,0).


%%%
hf_cis_connectivity(ethylenic,AA,BB,Sub,Connectivity):-
append(AA,BB,Subs1),
flatten(Subs1,Subs),
length(Subs,NumS),
hf_branch_ethylenic(pp,Val),
Connectivity is Val * NumS,
        hf_data(connectivity,Sub,Connectivity).
hf_cis_connectivity(oethylenic,AA,BB,Sub,Connectivity):-
append(AA,BB,Subs1Q),
flatten(Subs1Q,SubsQ),
findall(Name,atom_type(Name,q),ListQ),
difference(SubsQ,ListQ,Subs),
length(Subs,NumS),
hf_branch_oethylenic(pp,Val),
Connectivity is Val * NumS,
        hf_data(connectivity,Sub,Connectivity).
```

```
/*** hf_cis_connectivity_aux(oeth,SizesA1,ConA),
hf_cis_connectivity_aux(oeth,SizesB1,ConB),
Connectivity is ConA + ConB,
        hf_data(connectivity,Sub,Connectivity).
***/
hf_cis_connectivity(nethylenic,AA,BB,Sub,Connectivity):-
append(AA,BB,Subs1),
flatten(Subs1,Subs),
        findall(X,(member(X,Subs),
           substituent_type(_,X,Type),
                   \+ memberchk(Type,[neth])),NSubs),
length(NSubs,NumS),
hf_branch_nethylenic(pp,Val),
Connectivity is Val * NumS,
        hf_data(connectivity,Sub,Connectivity).
hf_cis_connectivity(sethylenic,AA,BB,Sub,Connectivity):-
append(AA,BB,Subs1),
flatten(Subs1,Subs),
        findall(X,(member(X,Subs),
           substituent_type(_,X,Type),
                   \+ memberchk(Type,[seth])),NSubs),
length(NSubs,NumS),
hf_branch_sethylenic(pp,Val),
Connectivity is Val * NumS,
        hf_data(connectivity,Sub,Connectivity).


hf_cis_connectivity_aux(oeth,[Size1,Size2],Con):-
hf_branch_oethylenic(pp,Val),
Con is (Size1 * Val) + (Size2 * Val).


hf_cis_aux2(AA,_,_,[0,0]) :-
flatten(AA,FlatA),
length(FlatA,LenA),
LenA = 0.
hf_cis_aux2(AA,A,_,Sizes) :-
flatten(AA,FlatA),
length(FlatA,LenA),
LenA = 1,
hf_cis_aux2_aux(A,AA,_Sub,Sizes).
hf_cis_aux2(AA,A,Sub,Sizes) :-
```

```
flatten(AA,FlatA),
length(FlatA,LenA),
LenA = 2,
FlatA = [X,Y],
pair(Sub,A,X,SubX),
pair(Sub,A,Y,SubY),
hf_get_sub_size(Sub,[SubX,SubY],Sizes).

hf_cis_aux2_aux(A,[[],[X]],Sub,Sizes) :-
pair(Sub,A,X,SubX),
hf_get_sub_size(Sub,[SubX],Size),
maximum([0,Size], Size1),
Sizes = [0,Size1].
hf_cis_aux2_aux(A,[[X],[]],Sub,Sizes) :-
pair(Sub,A,X,SubX),
hf_get_sub_size(Sub,[SubX],Size),
maximum([0,Size], Size1),
Sizes = [Size1,0].

hf_cis_1(SizesA,AA,Sub,Value) :-%when 2 subs on same carbon
flatten(SizesA,Flat),
delete(Flat,0,NewFlat),
length(NewFlat,LenNF),
LenNF > 1,

hf_cis_1_res(AA,0,Resonance),
        hf_data(resonance1,Sub,Resonance),
sumlist(SizesA,Size),
ESize is Size - (LenNF * 0.05),
maximum([0,ESize], EffectiveSize),
n_o_s_type(Sub,Type),
hf_cis_1_aux(Type,C),
        Steric is C*exp(EffectiveSize),
        hf_data(steric1,Sub,Steric),

        Value is Resonance + Steric,
!.
hf_cis_1(_,_,_,0).



/****************************************************************
Gets the proper multiplier depending on the type of
ethylenic
****************************************************************/
```

```
hf_cis_1_aux(ethylenic,C):-
hf_steric_ethylenic(pp,C).
hf_cis_1_aux(oethylenic,C):-
hf_steric_oethylenic(pp,C).
hf_cis_1_aux(nethylenic,C):-
hf_steric_nethylenic(pp,C).
hf_cis_1_aux(sethylenic,C):-
hf_steric_sethylenic(pp,C).


/*************************************************************
Gets the resonance value of groups off of an ethylenic
-I'm setting it to be -3.5582 = -7.1 for methyls
-for other groups I'll have to add to the code
*************************************************************/
hf_cis_1_res([],Out,Out).
hf_cis_1_res([[H]|T],Part,Resonance):-
reactophore_fact(R,_,AL,_,_),
memberchk(H,AL),
reac_type(R,_,Type),
%want to get for ANY connecting methyl group for thru res
hf_cis1_resonance(Type,Res),
NewPart is Part + Res,
hf_cis_1_res(T,NewPart,Resonance).
hf_cis_1_res([_|T],Part,Resonance):-
hf_cis_1_res(T,Part,Resonance).


/*************************************************************
Gets the correction for steric interaction of subs
on 'top' or 'bottom' of ethylenic
*************************************************************/
hf_cis_2([X,Y],[X1,Y1],[A,B],[A1,B1],Sub,Value) :-
flatten([X,B],Flat1),%deals with cis/Z
flatten([Y,A],Flat2),%deals with cis/Z
hf_cis_2_aux(Flat1,[X1,B1],V1),
hf_cis_2_aux(Flat2,[Y1,A1],V2),
Value is V1 + V2,
hf_data(steric2,Sub,Value),
!.
hf_cis_2(_,_,_,_,_,0).

/*****************************************************************
increases value exponetially as ESize increases.  Uses the full size
of the substituents so gets some kind of correction.
```

```
***********************************************************************/
hf_cis_2_aux([X,B],_,Value):-
delete([X,B],0,NewFlat),
length(NewFlat,LNF),
LNF > 1,
!,
ESizeX is X,
maximum([0,ESizeX],SX),
ESizeB is B,
maximum([0,ESizeB],SB),
ESize is SX + SB,
hf_steric_ethylenic_2(pp,C),
Value is C * exp(ESize).
hf_cis_2_aux(_,_,0).


/***********************************************************************
        Correction that depends on the length of the ring being looked at:
          'Len' = 3,4,5,6,7,8
           other is any other ring not of the specified length
***********************************************************************/
get_hf_ring_db(Len,_,Value) :-
Len < 8,
        hf_ring_db(Len,Value),
        !.
get_hf_ring_db(Len,Sub,Value) :-
Len >= 8,
reactophore_fact(Sub,e,_,_,[[A1,A2],[B1,B2]]),
get_hf_ring_db_aux(Len,A1,A2,B1,B2,Value),
        !.
get_hf_ring_db(_,_,V) :-
        hf_ring_db(other,V).

/************************************************************
gets the correction for rings that can have either
cis or trans in them.
the 4 'others' are the ending rule
************************************************************/
get_hf_ring_db_aux(Len,[],[_],[_],[],Value):-%cis
hf_ring_db(Len-cis,Value),
!.
get_hf_ring_db_aux(Len,[_],[],[],[_],Value):-%cis
hf_ring_db(Len-cis,Value),
!.
get_hf_ring_db_aux(Len,[],_,[],_,Value):-%trans
```

```
hf_ring_db(Len-trans,Value),
!.
get_hf_ring_db_aux(Len,_,[],_,[],Value):-%trans
hf_ring_db(Len-trans,Value),
!.
get_hf_ring_db_aux(_,[],[_],[_],[],Value):-%other
hf_ring_db(other-cis,Value),
!.
get_hf_ring_db_aux(_,[_],[],[],[_],Value):-%other
hf_ring_db(other-cis,Value),
!.
get_hf_ring_db_aux(_,[],_,[],_,Value):-%other
hf_ring_db(other-trans,Value),
!.
get_hf_ring_db_aux(_,_,[],_,[],Value):-%other
hf_ring_db(other-trans,Value),
!.

/**********************************************************************
Determines whether an ethylenic sub is a neth, oeth, seth, or eth
**********************************************************************/
n_o_s_type(Sub,Type):-
reactophore_fact(Sub,e,_,[A1,A2],_),
( A1 = 'O' -> Type = oethylenic
    ; (A1 = 'N' -> Type = nethylenic
 ; (A1 = 'S' -> Type = sethylenic
         ;
( A2 = 'O' -> Type = oethylenic
    ; (A2 = 'N' -> Type = nethylenic
 ; (A2 = 'S' -> Type = sethylenic
             ; Type = ethylenic
)))))).

/************************************************************************
        Asserts the pieces of the calculations into the database.
        These will get called later up in hf_values(_).
        They will be retracted at the top of the program
************************************************************************/
hf_data(Name,Reac,Value):-
        assert(data_tad(Name,Reac,Value)).

/*******************************************************
Finds all the substituents connected to Sub that have
pi orbitals
*******************************************************/
```

```
hf_pi_subs(Sub,ListPi):-
findall(XX,(pair(Sub,XX),
                    reac_type(XX,_,Pi),
                    memberchk(Pi,[aromatic,ethylenic,acetylenic])),ListPi).


/**** Data Area ****************************************************/

% This is now in a file called 'hf_data.pro'
```

C.2 HF_RING2.PRO

```
/****************************************************************
This module:
Finds all the special substructures in a molecule, eg
norbornane; 2 norbornanes; cubane; diadamantane, etc
Gets a correction for each of the substructures (and any branches
coming off of the substructure)
Finds the regular rings that are not part of the substructures
Gets a correction for branches off of the substructure

Gets a correction for how each of the substructures are
connected to each other.
****************************************************************/
hf_ring_structure_correct(RAs,AllRings,Value):-
hf_ring_structure(AllRings,StructuresVal),
hf_ring_find(_,MostStructures),
hf_ring_regular(AllRings,AllRings,MostStructures,[]),
hf_ring_find(_,AllSts),
hf_ring_structure_filter(AllSts,FilteredSts),
hf_ring_find(regular,RegRings),
hf_ring_regular2(RegRings,AllRings,0,RegRingsVal),
hf_ring_structure_connect(FilteredSts,FilteredSts,AllRings,RAs,0,
ConnectsVal),
Value is StructuresVal + RegRingsVal + ConnectsVal.

/****************************************************************
This module:

Finds all the substructures in a molecule
Gets the value for that structure and
any branches or
dbs part of or coming off of the structure
```

```
*************************************************************/
hf_ring_structure(AllRings,StructuresVal):-
hf_ring_bridge(AllRings,BridgeVal),
hf_ring_5553(AllRings,FiveVal),
hf_ring_adamantane(AllRings,AdaVal),
hf_ring_protoadamantane(AllRings,ProtoAdaVal),
hf_ring_tetrane(AllRings,TetraneVal),
hf_ring_cubane(AllRings,CubaneVal),
hf_ring_bullvalene(AllRings,BullvaleneVal),
hf_ring_bridgeane(AllRings,BridgeaneVal),
hf_ring_housane(AllRings,HousaneVal),
hf_ring_hatane(AllRings,HataneVal),
hf_ring_diadamantane(AllRings,DiaAdaVal),
hf_ring_33bridge(AllRings,TTBridgeVal),
hf_ring_azulene(AllRings,AzVal),

StructuresVal is BridgeVal + FiveVal + AdaVal + ProtoAdaVal +
         TetraneVal + CubaneVal + BullvaleneVal +
 BridgeaneVal + HousaneVal + HataneVal + DiaAdaVal +
 TTBridgeVal + AzVal.
hf_ring_structure(_,_):-
write('something died in hf_ring_structure'),nl,
abort.

/*****************************************************************
Each of these finds all of the predefined structures
in a molecule, and gets the appropriate correction for
each
*****************************************************************/
/*************************************************************
For bridges
*************************************************************/
hf_ring_bridge(AllRings,BridgeVal):-
/****
findall([R1,R2,R3],(member(R1,AllRings),
        member(R2,AllRings),
        R1 \= R2,
    intersects(R1,R2,[_,_,_|_]),
    member(R3,AllRings),
    R1 \= R3,
    R2 \= R3,
    intersects(R1,R3,[_,_,_|_]),
    intersects(R2,R3,[_,_,_|_]),
    hf_bridge_atom([R1,R2,R3],[_A,_B])
    ),List),
duplicate_list_sort(List,ListBridge),
hf_bridge_in_other(ListBridge,AllRings,ListBridgeInOther),
length(ListBridgeInOther,0),
hf_ring_structure_assert(ListBridge,bridge),
***/
hf_ring_find(bridge,ListBridge),
hf_ring_bridge_aux(ListBridge,AllRings,0,BridgeVal).
hf_ring_bridge(_,0).

/************************************************************
Finds if any of the bridges are also in two other rings,
where those two rings are connected to the edges where the
bridge atom is
************************************************************/
hf_bridge_in_other([],_,[]).
hf_bridge_in_other([[A,B,C]|Rest],AllRings,[[A,B,C]|ListBridge]):-
findall([X,Y],(member(X,AllRings),
       X \= A, X \= B, X \= C,
       intersects(A,X,[_,_]),
       intersects(B,X,[_,_]),
       intersects(C,X,[_,_,_]),
       member(Y,AllRings),
       Y \= A, Y \= B, Y \= C,
       X \= Y,
       intersects(A,Y,[_,_]),
       intersects(B,Y,[_,_]),
       intersects(C,Y,[_,_,_])),[_|_]),
hf_bridge_in_other(Rest,AllRings,ListBridge).
hf_bridge_in_other([_H|Rest],AllRings,ListBridge):-
hf_bridge_in_other(Rest,AllRings,ListBridge).

/************************************************************
Gets the correction based on the size of the bridge and
the double bonds within it and
the branches coming off
************************************************************/
hf_ring_bridge_aux([],_,Val,Val).
hf_ring_bridge_aux([H|T],AllRings,Part,Out):-
gensort(H,longer,[A,B,C]),
length(A,LenA),
length(B,LenB),
length(C,LenC),
concatenate_atomlist_to_atom([LenA,LenB,LenC],Key),
hf_bridge(Key,Val),
hf_bridge_atom(H,Atoms),
%%trace,
hf_ring_5553_aux_branch(H,AllRings,Atoms,0,BranchCorrect),
hf_ring_bridge_aux_db(H,Atoms,0,DbCorrect),
NewPart is Part + Val + BranchCorrect + DbCorrect,
hf_ring_bridge_aux(T,AllRings,NewPart,Out).
hf_ring_bridge_aux([_H|T],AllRings,Part,Out):-
hf_ring_bridge_aux(T,AllRings,Part,Out).

/*************************************************************
Takes an atom,
finds the atoms which are not part of the structure;
makes sure the Atom is not in a ring that is not part
of the structure
finds the other atoms hooked to Atom
find those others that are not part of the SAs
*************************************************************/
hf_rba_branch_aux_aux(Atom,H,SAs,AllRings,ListOut):-
difference(AllRings,H,RestRings),
flatten(RestRings,FRR1),
delete_duplicates(FRR1,FRR),
\+ memberchk(Atom,FRR),
atom_specs(_,Atom,_,_,List),
     findall(Y,(member(Y,List),
            \+ memberchk(Y,SAs)), ListOut).


************************************************************
Finds any endo ethylenics in a structure and
which ring they are in (the short one)
passes this on
************************************************************/
hf_ring_bridge_aux_db(H,Atoms,_Part,Val):-
findall([X,Ring],(reactophore_fact(X,e,[E1,E2],_,Hooks),
      flatten(Hooks,_FH),
      hf_which_ring(E1,H,Ring),
      memberchk(E1,Ring),
      memberchk(E2,Ring)),ListE1),
delete_duplicates(ListE1,ListE),
hf_rba_db_aux(ListE,Atoms,0,Val).

/************************************************************
Takes the list of ethylenics
1.  For those e's that BOTH hooks are connected to bridge atoms
    get value based on size of ring the 'e' is in
2.  For those e's in a 6-membered ring who has 1 hook connected to
```

```
    a bridge atom
3.  Doesn't fit either of these rules, just ignore and recall
*************************************************************/
hf_rba_db_aux([],_,Out,Out).
hf_rba_db_aux([[E,Ring]|T],[BA1,BA2],Part,V):-
reactophore_fact(E,e,_,_,Hooks),
flatten(Hooks,FH),
memberchk(BA1,FH),
memberchk(BA2,FH),
length(Ring,LenRing),
hf_bridgedb(LenRing,Val),
NewPart is Part + Val,
hf_rba_db_aux(T,[BA1,BA2],NewPart,V).
hf_rba_db_aux([[E,Ring]|T],[BA1,BA2],Part,V):-
length(Ring,6),
reactophore_fact(E,e,_,_,Hooks),
flatten(Hooks,FH),
(memberchk(BA1,FH), \+ memberchk(BA2,FH) ;
 \+ memberchk(BA1,FH), memberchk(BA2,FH)),
hf_bridgedb(b6,Val),
NewPart is Part + Val,
hf_rba_db_aux(T,[BA1,BA2],NewPart,V).
hf_rba_db_aux([_|T],BAs,Part,V):-
hf_rba_db_aux(T,BAs,Part,V).

/*************************************************************
For 5553 molecules
*************************************************************/
hf_ring_5553(AllRings,FiveVal):-
/*******
findall([R1,R2,R3,R4],
(member(R1,AllRings),
 length(R1,3),
 member(R2,AllRings),
 length(R2,5),
 member(R3,AllRings),
 length(R3,5),
         R2 \= R3,
 member(R4,AllRings),
 length(R4,5),
         R2 \= R4,
         R3 \= R4,
 intersects(R1,R2,[_,_]),
 intersects(R1,R3,[_,_]),
 intersects(R1,R4,[_,_]),
 intersects(R2,R3,[_,_,_]),
 intersects(R2,R4,[_,_,_]),
 intersects(R3,R4,[_,_,_])),List),
duplicate_list_sort(List,ListFive),
hf_ring_structure_assert(ListFive,5553),
*******/
hf_ring_find(5553,ListFive),
hf_ring_5553_aux(ListFive,AllRings,0,FiveVal).

hf_ring_5553_aux([],_,Out,Out).
hf_ring_5553_aux([H|T],AllRings,In,Out):-
hf_bridge(5553,Val),
hf_ring_5553_auxdb(H,DbCorrect),
hf_atom_in3rings(H,Atoms),
hf_ring_5553_aux_branch(H,AllRings,Atoms,0,BranchCorrect),
NewIn is In + Val + DbCorrect + BranchCorrect,
hf_ring_5553_aux(T,AllRings,NewIn,Out).

/***************************************************************
None w/ dbs so don't know how to model
***************************************************************/
hf_ring_5553_auxdb(_,0).

/***************************************************************
Gets the correction for branches coming off of the atoms in
the structure, if nothing coming off then get no correction
***************************************************************/
hf_ring_5553_aux_branch(H,AllRings,Atoms,0,BranchCorrect):-
flatten(H,SAs1),
delete_duplicates(SAs1,SAs),
hf_r5553a_branch_aux(SAs,H,SAs,AllRings,Atoms,0,BranchCorrect).

/***************************************************************
This module handles bridge atoms and atoms that are in three rings
where 2 of those rings share 1 side w/ the 3rd ring
and it also handles branches coming off of regular rings

Takes each atom

finds the atoms connected to it
gets those that are not members of the current structure
gets the rings that are not members of the current structure
makes sure that the atom is not in another ring
finds the number of branches coming off the atom
```

```
if atom is in three rings get one correction
may need to add special correction for being in 553
else get other correction
************************************************************/
hf_r5553a_branch_aux([],_,_,_,_,Out,Out).
hf_r5553a_branch_aux([Atom|Rest],H,SAs,AllRings,Atoms,Part,BC):-
hf_rba_branch_aux_aux(Atom,H,SAs,AllRings,LO),
length(LO,Len),
hf_ba_connect(Atom,Atoms,BAC),
(memberchk(Atom,Atoms) ->
((findall([Ring1,Ring2,Ring3],
  (member(Ring1,H),
   member(Ring2,H),
   Ring1 \= Ring2,
   member(Ring3,H),
   Ring1 \= Ring3,
   Ring2 \= Ring3,
   memberchk(Atom,Ring1),
   memberchk(Atom,Ring2),
   memberchk(Atom,Ring3),
       intersects(Ring1,Ring3,[_,_]),
             intersects(Ring2,Ring3,[_,_]),
       intersects(Ring1,Ring2,[_,_|_])),[_|_])
  )  -> ring_vertices_3ring(Len,V)
       ; ring_vertices_3ring(Len,V)
)
;
(BAC =:= 1 -> hf_ba_connect2(Atom,H,Len,AllRings,V)
    ; ring_vertices_1ring(Len,V)
)
),
NewPart is Part + V,
hf_r5553a_branch_aux(Rest,H,SAs,AllRings,Atoms,NewPart,BC).
hf_r5553a_branch_aux([_|Rest],H,SAs,AllRings,Atoms,NewPart,BC):-
hf_r5553a_branch_aux(Rest,H,SAs,AllRings,Atoms,NewPart,BC).


/************************************************************
Checks that Atom (atom in the ring) is connected to at least
two bridge atoms - if so then BAC = 1 - else BAC = 0
************************************************************/
hf_ba_connect(Atom,Atoms,1):-
findall([BA1,BA2],(member(BA1,Atoms),
   bonded(BA1,_,Atom),
   member(BA2,Atoms),
   BA2 \= BA1,
   bonded(BA2,_,Atom)),[_|_]).
hf_ba_connect(_,_,0).

/***************************************************************
Finds what kind of Atom it is - if 'ethylenic' and
in two five membered rings get special correction.
correction, else - get regular correction
***************************************************************/
hf_ba_connect2(Atom,H,_Len,AllRings,V):-
reac_type(Atom,ethylenic),
hf_ba_connect2_aux(Atom,H,AllRings,V).
hf_ba_connect2(_,_,Len,_,V):-
ring_vertices_1ring(Len,V).

hf_ba_connect2_aux(Atom,_H,_AllRings,V):-
findall(Sub,(reactophore_fact(Sub,e,Parts,_,_),
     memberchk(Atom,Parts)),[Sub]),
neth_oeth_seth(Sub),
n_o_s_type(Sub,Type),
hf_ba_connect2_aux_aux(Type,V1),
ring_vertices_2ring(bridge2,Real),
V is Real-V1.
hf_ba_connect2_aux(_Atom,_H,_AllRings,V):-
hf_ring_exo(5,V1),
ring_vertices_2ring(bridge2,Real),
V is Real-V1.

/******************************
gets the appropriate correction
******************************/
hf_ba_connect2_aux_aux(oethylenic,V1):-
hf_ring_exo_oethylenic(5,V1).
hf_ba_connect2_aux_aux(_,0).


/***************************************************************
For adamantane
***************************************************************/
hf_ring_adamantane(AllRings,AdaVal):-
/****
hf_ring_adamantane_aux2(AllRings,ListAda),
\+ hf_ada_share(ListAda,ListAda,0),
hf_ring_structure_assert(ListAda,adamantane),
*****/
```

```
hf_ring_find(adamantane,ListAda),
hf_ring_adamantane_aux(ListAda,AllRings,0,AdaVal).
hf_ring_adamantane(_,0).

hf_ring_adamantane_aux([],_,Out,Out).
hf_ring_adamantane_aux([H|T],AllRings,In,Out):-
hf_bridge(adamantane,Val),
hf_ring_adamantane_auxdb(H,DbCorrect),
hf_atom_in3rings(H,Atoms),
hf_ring_ada_aux_branch(H,AllRings,Atoms,[],0,BranchCorrect),
NewIn is In + Val + DbCorrect + BranchCorrect,
hf_ring_adamantane_aux(T,AllRings,NewIn,Out).

/****************************************************************
None w/ dbs so don't know how to model
****************************************************************/
hf_ring_adamantane_auxdb(_,0).

/****************************************************************
Gets the correction for branches coming off of the atoms in
the structure, if nothing coming off then get no correction
****************************************************************/
hf_ring_ada_aux_branch(H,AllRings,Atoms,Atoms4,0,BC):-
flatten(H,SAs1),
delete_duplicates(SAs1,SAs),
hf_radaa_branch_aux(SAs,H,SAs,AllRings,Atoms,Atoms4,0,BC).

/****************************************************************
Takes each atom
finds the number of branches coming off the atom

if atom is in 3 rings
if it's also in 4 rings
if it is then get an ad4 value
 else get an ad3 value
else get a value for just being in 1 ring

****************************************************************/

hf_radaa_branch_aux([],_,_,_,_,_,Out,Out).
hf_radaa_branch_aux([Atom|Rest],H,SAs,AllRings,Atoms,Atoms4,Part,BC):-
hf_rba_branch_aux_aux(Atom,H,SAs,AllRings,LO),
length(LO,Len),

(memberchk(Atom,Atoms) ->
(memberchk(Atom,Atoms4) -> ring_vertices_4ring(ad-Len,V)
                 ; ring_vertices_3ring(ad-Len,V)
)
; ring_vertices_1ring(Len,V)
),
NewPart is Part + V,
hf_radaa_branch_aux(Rest,H,SAs,AllRings,Atoms,Atoms4,NewPart,BC).
hf_radaa_branch_aux([_Atom|Rest],H,SAs,AllRings,Atoms,Atoms4,Part,BC):-
hf_radaa_branch_aux(Rest,H,SAs,AllRings,Atoms,Atoms4,Part,BC).


/************************************************************
Finds all those sets that are joined together like
adamantane
************************************************************/
hf_ring_adamantane_aux2(AllRings,ListAda):-
findall([R1,R2,R3,R4],
(member(R1,AllRings),
 length(R1,6),
 member(R2,AllRings),
 length(R2,6),
 R1 \= R2,
 member(R3,AllRings),
 length(R3,6),
 R1 \= R3,
         R2 \= R3,
 member(R4,AllRings),
 length(R4,6),
 R1 \= R4,
         R2 \= R4,
         R3 \= R4,
 intersects(R1,R2,[_,_,_]),
 intersects(R1,R3,[_,_,_]),
 intersects(R1,R4,[_,_,_]),
 intersects(R2,R3,[_,_,_]),
 intersects(R2,R4,[_,_,_]),
 intersects(R3,R4,[_,_,_])),List),
duplicate_list_sort(List,ListAda).


/************************************************************
Checks to see if the list of adamantanes have a ring in common
If they do, the '\+' above causes it to get kicked out and
is not considered adamantane
************************************************************/
```

```
hf_ada_share([],[],_):- fail.
hf_ada_share([_],[_],_):- fail.
hf_ada_share([],_ListAda,Num):-
(Num =:= 0 -> fail
    ; true
).
hf_ada_share([H|T],ListAda,In):-
findall(_X,(member(Other,ListAda),
   Other \= H,
   intersects(H,Other,[_|_])),List),
length(List,Num),
NewIn is In + Num,
hf_ada_share(T,ListAda,NewIn).


/*****************************************************************
For protoadamantane
*****************************************************************/
hf_ring_protoadamantane(AllRings,ProtoAdaVal):-
/********
findall([R1,R2,R3,R4],
(member(R1,AllRings),
 length(R1,6),
 member(R2,AllRings),
 length(R2,6),
 R1 \= R2,
 member(R3,AllRings),
 length(R3,5),
 member(R4,AllRings),
 length(R4,7),
 intersects(R1,R2,[_,_,_]),
 intersects(R1,R3,[_,_]),
 intersects(R1,R4,[_,_,_,_]),
 intersects(R2,R3,[_,_,_]),
 intersects(R2,R4,[_,_,_]),
 intersects(R3,R4,[_,_,_])),List),
duplicate_list_sort(List,ListProto),
hf_ring_structure_assert(ListProto,protoadamantane),
*******/
hf_ring_find(protoadamantane,ListProto),
hf_ring_protoadamantane_aux(ListProto,AllRings,0,ProtoAdaVal).

hf_ring_protoadamantane_aux([],_,Out,Out).
hf_ring_protoadamantane_aux([H|T],AllRings,In,Out):-
hf_bridge(protoadamantane,Val),
hf_ring_protoadamantane_auxdb(H,DbCorrect),
hf_atom_in3rings(H,Atoms),
hf_ring_ada_aux_branch(H,AllRings,Atoms,[],0,BranchCorrect),
NewIn is In + Val + DbCorrect + BranchCorrect,
hf_ring_protoadamantane_aux(T,AllRings,NewIn,Out).

/***************************************************************
No structures w/ dbs
***************************************************************/
hf_ring_protoadamantane_auxdb(_,0).

/***************************************************************
For tetrane
***************************************************************/
hf_ring_tetrane(AllRings,TetraneVal):-
/*********
findall([R1,R2,R3,R4],
(member(R1,AllRings),
 length(R1,3),
 member(R2,AllRings),
 length(R2,3),
 R1 \= R2,
 member(R3,AllRings),
 length(R3,3),
 R1 \= R3,
         R2 \= R3,
 member(R4,AllRings),
 length(R4,3),
 R1 \= R4,
         R2 \= R4,
         R3 \= R4,
 intersects(R1,R2,[_,_]),
 intersects(R1,R3,[_,_]),
 intersects(R1,R4,[_,_]),
 intersects(R2,R3,[_,_]),
 intersects(R2,R4,[_,_]),
 intersects(R3,R4,[_,_])),List),
duplicate_list_sort(List,ListTetrane),
hf_ring_structure_assert(ListTetrane,tetrane),
***********/
hf_ring_find(tetrane,ListTetrane),
hf_ring_tetrane_aux(ListTetrane,AllRings,0,TetraneVal).

hf_ring_tetrane_aux([],_,Out,Out).
hf_ring_tetrane_aux([H|T],AllRings,In,Out):-
```

```
hf_bridge(tetrane,Val),
hf_ring_tetrane_auxdb(H,DbCorrect),
hf_atom_in3rings(H,Atoms),
%%% hf_ring_5553_aux_branch(H,AllRings,Atoms,0,BranchCorrect),
hf_ring_ada_aux_branch(H,AllRings,Atoms,[],0,BranchCorrect),
NewIn is In + Val + DbCorrect + BranchCorrect,
hf_ring_tetrane_aux(T,AllRings,NewIn,Out).

/*************************************************************
None w/ db's now
*************************************************************/
hf_ring_tetrane_auxdb(_,0).


/*************************************************************
For cubane
*************************************************************/
hf_ring_cubane(AllRings,CubaneVal):-
/*******
findall([R1,R2,R3,R4,R5,R6],
(member(R1,AllRings),
 length(R1,4),
 member(R2,AllRings),
 length(R2,4),
 R1 \= R2,
 member(R3,AllRings),
 length(R3,4),
 R1 \= R3,
         R2 \= R3,
 member(R4,AllRings),
 length(R4,4),
 R1 \= R4,
         R2 \= R4,
         R3 \= R4,
 member(R5,AllRings),
 length(R5,4),
 R1 \= R5,
         R2 \= R5,
         R3 \= R5,
 R4 \= R5,
 member(R6,AllRings),
 length(R6,4),
 R1 \= R6,
         R2 \= R6,
         R3 \= R6,
 R4 \= R6,
 R5 \= R6,
 findall([A,B],(member(A,[R1,R2,R3,R4,R5,R6]),
        member(B,[R1,R2,R3,R4,R5,R6]),
A \= B,
intersects(A,B,[_,_])),ListAB),
 delete_duplicates(ListAB,AB),
 length(AB,24)),List),
duplicate_list_sort(List,ListCubane),
hf_ring_structure_assert(ListCubane,cubane),
********/
hf_ring_find(cubane,ListCubane),
hf_ring_cubane_aux(ListCubane,AllRings,0,CubaneVal).

hf_ring_cubane_aux([],_,Out,Out).
hf_ring_cubane_aux([H|T],AllRings,In,Out):-
hf_bridge(cubane,Val),
hf_ring_cubane_auxdb(H,DbCorrect),
hf_atom_in3rings(H,Atoms),
hf_ring_5553_aux_branch(H,AllRings,Atoms,0,BranchCorrect),
NewIn is In + Val + DbCorrect + BranchCorrect,
hf_ring_cubane_aux(T,AllRings,NewIn,Out).

/*************************************************************
No db's right now
*************************************************************/
hf_ring_cubane_auxdb(_,0).

/*************************************************************
For bullvalene
right now we only have bullvalene (which already has
dbs in it, so until we get one that doesn't, we're
not going to have a separate db correction)
*************************************************************/
hf_ring_bullvalene(AllRings,BullaVal):-
/*********
findall([R1,R2,R3,R4],
(member(R1,AllRings),
 length(R1,3),
 member(R2,AllRings),
 length(R2,7),
 member(R3,AllRings),
 length(R3,7),
         R2 \= R3,
 member(R4,AllRings),
 length(R4,7),
         R2 \= R4,
         R3 \= R4,
 intersects(R1,R2,[_,_]),
 intersects(R1,R3,[_,_]),
 intersects(R1,R4,[_,_]),
 intersects(R2,R3,[_,_,_,_]),
 intersects(R2,R4,[_,_,_,_]),
 intersects(R3,R4,[_,_,_,_])),List),
duplicate_list_sort(List,ListBulla),
hf_ring_structure_assert(ListBulla,bullvalene),
*********/
hf_ring_find(bullvalene,ListBulla),
hf_ring_bullvalene_aux(ListBulla,AllRings,0,BullaVal).

hf_ring_bullvalene_aux([],_,Out,Out).
hf_ring_bullvalene_aux([H|T],AllRings,In,Out):-
hf_bridge(bullvalene,Val),
hf_ring_bullvalene_auxdb(H,DbCorrect),
hf_atom_in3rings(H,Atoms),
hf_ring_5553_aux_branch(H,AllRings,Atoms,0,BranchCorrect),
NewIn is In + Val + DbCorrect + BranchCorrect,
hf_ring_bullvalene_aux(T,AllRings,NewIn,Out).

hf_ring_bullvalene_auxdb(_,0).

/***************************************************************
For bridgeane
***************************************************************/
hf_ring_bridgeane(AllRings,BridgeaneVal):-
/*********
findall([R1,R2,R3,R4,R5],
(member(R1,AllRings),
 length(R1,4),
 member(R2,AllRings),
 length(R2,3),
 member(R3,AllRings),
 length(R3,3),
         R2 \= R3,
 member(R4,AllRings),
 length(R4,5),
 member(R5,AllRings),
 length(R5,5),
         R4 \= R5,
 intersects(R1,R2,[_,_]),
 intersects(R1,R3,[_,_]),
 intersects(R1,R4,[_,_]),
 intersects(R1,R5,[_,_]),
 intersects(R2,R3,[]),
 intersects(R2,R4,[_,_]),
 intersects(R2,R5,[_,_]),
 intersects(R3,R4,[_,_]),
 intersects(R3,R5,[_,_])),List),
duplicate_list_sort(List,ListBridgeane),
hf_ring_structure_assert(ListBridgeane,bridgeane),
*********/
hf_ring_find(bridgeane,ListBridgeane),
hf_ring_bridgeane_aux(ListBridgeane,AllRings,0,BridgeaneVal).

hf_ring_bridgeane_aux([],_,Out,Out).
hf_ring_bridgeane_aux([H|T],AllRings,In,Out):-
hf_bridge(bridgeane,Val),
hf_ring_bridgeane_auxdb(H,DbCorrect),
hf_atom_in3rings(H,Atoms),
hf_ring_5553_aux_branch(H,AllRings,Atoms,0,BranchCorrect),
NewIn is In + Val + DbCorrect + BranchCorrect,
hf_ring_bridgeane_aux(T,AllRings,NewIn,Out).

hf_ring_bridgeane_auxdb(_,0).

/***************************************************************
For housane
***************************************************************/
hf_ring_housane(AllRings,HousaneVal):-
/************
        findall(RL4,(member(RL4,AllRings),
                     length(RL4,4)),List4s),
        findall(RL5,(member(RL5,AllRings),
                     length(RL5,5)),List5s),
        findall(RL6,(member(RL6,AllRings),
                     length(RL6,6)),List6s),
        findall([R1,R2,R3,R4,R5,R6,R7,R8,R9],
                        (member(R1,List4s),
                         member(R2,List5s),
                         member(R3,List5s),
                         R2 \= R3,
                         member(R4,List5s),
                         R2 \= R4,
                         R3 \= R4,
                         member(R5,List5s),
                         R2 \= R5,
                         R3 \= R5,
                         R4 \= R5,
                         member(R6,List5s),
                         R2 \= R6,
                         R3 \= R6,
                         R4 \= R6,
                         R5 \= R6,
                         member(R7,List5s),
                         R2 \= R7,
                         R3 \= R7,
                         R4 \= R7,
                         R5 \= R7,
                         R6 \= R7,
                         member(R8,List6s),
                         member(R9,List6s),
                         R8 \= R9,
 findall(X,(member(X,List5s),
    intersects(R1,X,[_,_])),List4),
 findall(Q,(member(Q,List6s),
    member(QQ,List4),
    intersects(QQ,Q,[_,_,_|_])),List3),
 length(List3,6),
 difference(List5s,List4,[A,B]),
 intersects(A,B,[_,_])),List),
duplicate_list_sort(List,ListHouse),
hf_ring_structure_assert(ListHouse,houseane),
************/
hf_ring_find(houseane,ListHouse),
hf_ring_housane_aux(ListHouse,AllRings,0,HousaneVal).
%used to not have AllRings 5-11-04
%could be correct
hf_ring_housane_aux([],_,Out,Out).
hf_ring_housane_aux([H|T],AllRings,In,Out):-
hf_bridge(housane,Val),
hf_ring_housane_auxdb(H,DbCorrect),
hf_atom_in3rings(H,Atoms),
hf_ring_5553_aux_branch(H,AllRings,Atoms,0,BranchCorrect),
NewIn is In + Val + DbCorrect + BranchCorrect,
hf_ring_housane_aux(T,AllRings,NewIn,Out).

hf_ring_housane_auxdb(_,0).


/**************************************************************
```

```
For hatane
**************************************************************/
hf_ring_hatane(AllRings,HataneVal):-
/**************
findall([R1,R2,R3,R4,R5],
(member(R1,AllRings),
 length(R1,6),
 member(R2,AllRings),
 length(R2,3),
 member(R3,AllRings),
 length(R3,3),
         R2 \= R3,
 member(R4,AllRings),
 length(R4,5),
 member(R5,AllRings),
 length(R5,5),
         R4 \= R5,
 intersects(R1,R2,[_,_]),
 intersects(R1,R3,[_,_]),
 intersects(R1,R4,[_,_,_]),
 intersects(R1,R5,[_,_,_]),
 intersects(R2,R3,[]),
 intersects(R4,R5,[_,_])),List),
duplicate_list_sort(List,ListHatane),
hf_ring_structure_assert(ListHatane,hatane),
************/
hf_ring_find(hatane,ListHatane),
hf_ring_hatane_aux(ListHatane,AllRings,0,HataneVal).

hf_ring_hatane_aux([],_AllRings,Out,Out).
hf_ring_hatane_aux([H|T],AllRings,In,Out):-
hf_bridge(hatane,Val),
hf_ring_hatane_auxdb(H,DbCorrect),
hf_atom_in3rings(H,Atoms),
hf_ring_5553_aux_branch(H,AllRings,Atoms,0,BranchCorrect),
NewIn is In + Val + DbCorrect + BranchCorrect,
hf_ring_hatane_aux(T,AllRings,NewIn,Out).

hf_ring_hatane_auxdb(_,0).

/**************************************************************
For diadamantane
**************************************************************/
hf_ring_diadamantane(AllRings,DiAdaVal):-
/**********
```

```
hf_ring_adamantane_aux2(AllRings,ListAda),
hf_ada_share(ListAda,ListAda,0),
findall(NewA,(member(A1,ListAda),
 member(A2,ListAda),
 A1 \= A2,
 intersects(A1,A2,[_]),
 append(A1,A2,NewA)),List),
duplicate_list_sort(List,ListDiAda),
hf_ring_structure_assert(ListDiAda,diadamantane),
************/
hf_ring_find(diadamantane,ListDiAda),
hf_ring_diadamantane_aux(ListDiAda,AllRings,0,DiAdaVal).
hf_ring_diadamantane(_,0).

hf_ring_diadamantane_aux([],_,Out,Out).
hf_ring_diadamantane_aux([H|T],AllRings,In,Out):-
hf_bridge(diadamantane,Val),
hf_ring_diadamantane_auxdb(H,DbCorrect),
hf_atom_in3rings(H,Atoms),
hf_atom_in4rings(H,Atoms4),
hf_ring_ada_aux_branch(H,AllRings,Atoms,Atoms4,0,BranchCorrect),
NewIn is In + Val + DbCorrect + BranchCorrect,
hf_ring_diadamantane_aux(T,AllRings,NewIn,Out).

hf_ring_diadamantane_auxdb(_,0).


/************************************************************
For 3-3 sharing a side and connected on each end
through a "bridge"
************************************************************/
hf_ring_33bridge(AllRings,TTBridgeVal):-
/************
findall([R1,R2,R3,R4],
(member(R1,AllRings),
 length(R1,3),
 member(R2,AllRings),
 length(R2,3),
        R1 \= R2,
 intersects(R1,R2,[_,_]),

 member(R3,AllRings),
 length(R3,L),
 member(R4,AllRings),
 length(R4,L),
        R3 \= R4,
 intersects(R3,R4,[_,_,_|_]),
 intersects(R1,R3,I13),
 length(I13,2),
 intersects(R2,R3,I23),
        length(I23,2),
        intersects(I13,I23,[_]),
 intersects(R1,R4,I14),
 length(I14,2),
 intersects(R2,R4,I24),
 length(I24,2),
        intersects(I13,I23,[_]),

        intersects(I13,I14,[_]),
        intersects(I23,I24,[_])),List),
duplicate_list_sort(List,ListTTBridge),
hf_ring_structure_assert(ListTTBridge,a33bridge),
*************/
hf_ring_find(a33bridge,ListTTBridge),

hf_ring_33bridge_aux(ListTTBridge,AllRings,0,TTBridgeVal).

hf_ring_33bridge_aux([],_,Out,Out).
hf_ring_33bridge_aux([H|T],AllRings,In,Out):-
hf_bridge(a33bridge,Val),
hf_ring_33bridge_auxdb(H,DbCorrect),
hf_atom_in3rings(H,Atoms),
hf_ring_33bridge_aux_branch(H,AllRings,Atoms,_,0,BranchCorrect),
NewIn is In + Val + DbCorrect + BranchCorrect,
hf_ring_33bridge_aux(T,AllRings,NewIn,Out).

/*******************************************************************
No db's right now
*******************************************************************/
hf_ring_33bridge_auxdb(_,0).

/*******************************************************************
Gets the correction for branches off of this structure
*******************************************************************/
hf_ring_33bridge_aux_branch(H,AllRings,Atoms,_,0,BC):-
flatten(H,SAs1),
delete_duplicates(SAs1,SAs),
hf_r33b_branch_aux(SAs,H,SAs,AllRings,Atoms,_,0,BC).

hf_r33b_branch_aux([],_,_,_,_,_,Out,Out).
```

```
hf_r33b_branch_aux([Atom|Rest],H,SAs,AllRings,Atoms,_,Part,BranchCorrect):-
hf_rba_branch_aux_aux(Atom,H,SAs,AllRings,LO),
length(LO,Len),

(memberchk(Atom,Atoms) ->
(findall([R1,R2,R3],
(member(R1,AllRings),
         length(R1,3),
         memberchk(Atom,R1),
         member(R2,AllRings),
         R1 \= R2,
         length(R2,3),
         memberchk(Atom,R2),
          member(R3,AllRings),
         memberchk(Atom,R3),
         length(R3,5)), [_,_]) -> ring_vertices_2ring(Len,V)
        ; ring_vertices_3ring(Len,V)
)

; ring_vertices_1ring(Len,V)
),
NewPart is Part + V,
hf_r33b_branch_aux(Rest,H,SAs,AllRings,Atoms,_Atoms4,NewPart,BranchCorrect).
hf_r33b_branch_aux([_Atom|Rest],H,SAs,AllRings,
Atoms,Atoms4,Part,BranchCorrect):-
hf_r33b_branch_aux(Rest,H,SAs,AllRings,Atoms,Atoms4,Part,BranchCorrect).


/**************************************************************
For azulenes
**************************************************************/
hf_ring_azulene(AllRings,AzVal):-
/***********
findall([R1,R2],(member(R1,AllRings),
 length(R1,5),
 hf_numes(R1,Num5Es),
         Num5Es =:= 2,
 member(R2,AllRings),
 length(R2,7),
 hf_numes(R2,Num7Es),
 Num7Es =:= 3,
 intersects(R1,R2,[_,_])),List),
duplicate_list_sort(List,ListAz),
hf_ring_structure_assert(ListAz,azulene),
************/
```

```
hf_ring_find(azulene,ListAz),
hf_ring_azulene_aux(ListAz,AllRings,0,AzVal).

hf_ring_azulene_aux([],_,Out,Out).
hf_ring_azulene_aux([_H|T],AllRings,In,Out):-
hf_bridge(azulene,Val),
NewIn is In + Val,
hf_ring_azulene_aux(T,AllRings,NewIn,Out).

/***********************************************************
This module finds the regular rings in a molecule and asserts
them in the database.

The basic correction for them was already done in hf.pro
***********************************************************/
hf_ring_regular([],AllRings,_,RingsInStructures):-
difference(AllRings,RingsInStructures,RegRings),
hf_rsa_reg(RegRings).
hf_ring_regular([Ring|Rest],AllRings,Structures,In):-
hf_ring_regular_aux(Ring,Structures,[],RIS),
append([RIS],In,NewIn),
hf_ring_regular(Rest,AllRings,Structures,NewIn).

hf_rsa_reg([]).
hf_rsa_reg([H|T]):-
hf_ring_structure_assert([[H]],regular),
hf_rsa_reg(T).

hf_ring_regular_aux(_,[],In,In).
hf_ring_regular_aux(Ring,[S1|Structures],In,Return):-
memberchk(Ring,S1),
append(Ring,In,NewIn),
hf_ring_regular_aux(Ring,Structures,NewIn,Return).
hf_ring_regular_aux(Ring,[_S1|RestSs],In,Return):-
hf_ring_regular_aux(Ring,RestSs,In,Return).

/***********************************************************
Gets the correction for the branches off of the
atoms in a regular ring
***********************************************************/
hf_ring_regular2([],_,Part,Part).
hf_ring_regular2([[H]|T],AllRings,Part,RegRingsVal):-
hf_ring_r2_aux(H,AllRings,[],0,Val),
NewPart is Part + Val,
hf_ring_regular2(T,AllRings,NewPart,RegRingsVal).
```

```
hf_ring_r2_aux(H,AllRings,Atoms,_,BC):-
flatten(H,SAs1),
delete_duplicates(SAs1,SAs),
hf_rr2a_branch_aux(SAs,H,SAs,AllRings,Atoms,0,BC).

hf_rr2a_branch_aux([],_,_,_,_,Out,Out).
hf_rr2a_branch_aux([Atom|Rest],H,SAs,AllRings,Atoms,Part,BC):-
hf_rba_branch_aux_aux(Atom,[H],SAs,AllRings,LO),
length(LO,Len),
hf_ring_branch_halo(LO,Atom,AllRings,Mult),
ring_vertices_1ring(Len,V),
NewPart is Part + V*Mult,
hf_rr2a_branch_aux(Rest,H,SAs,AllRings,Atoms,NewPart,BC).
hf_rr2a_branch_aux([_Atom|Rest],H,SAs,AllRings,Atoms,Part,BC):-
hf_rr2a_branch_aux(Rest,H,SAs,AllRings,Atoms,Part,BC).


/*************************************************************
This module takes the substituents off a particular atom
and gets the multiplier depending on how many are NOT methyl
*************************************************************/
hf_ring_branch_halo([Atom],_,_,Mult):-
reactophore_fact(T,_,Atoms,_,_),
memberchk(Atom,Atoms),
reac_type(T,_,Type),
ring_vertices_1ring(Type,Mult).
hf_ring_branch_halo([Atom1,Atom2],_,_,1):-
reactophore_fact(T1,_,Atoms1,_,_),
memberchk(Atom1,Atoms1),
reac_type(T1,_,methyl),
reactophore_fact(T2,_,Atoms2,_,_),
memberchk(Atom2,Atoms2),
reac_type(T2,_,methyl).
hf_ring_branch_halo([Atom1,Atom2],A,ARs,Mult):-
hf_ring_branch_halo([Atom1],_,_,M1),
hf_ring_branch_halo([Atom2],_,_,M2),
hf_ring_branch_halo_aux([Atom1,Atom2],A,ARs,M3),
Mult is M1 + M2 + M3.
hf_ring_branch_halo(_,_,_,0).

hf_ring_branch_halo_aux([Atom1,Atom2],A,ARs,M3):-
member(Ring,ARs),
length(Ring,3),
memberchk(A,Ring),
hf_halo_get_halo(Atom1,f),
hf_halo_get_halo(Atom2,f),
ring_vertices_1ring(ff3,M3).
hf_ring_branch_halo_aux([Atom1,Atom2],_,_,M3):-
hf_halo_get_halo(Atom1,f),
hf_halo_get_halo(Atom2,f),
ring_vertices_1ring(ff,M3).

/**************************************************************
This module finds the connections between substructures and
then corrects for that connection
**************************************************************/
hf_ring_structure_connect([],_,_,_,Out,Out).
hf_ring_structure_connect([StrX|Rest],AllSt,AllRings,RAs,Part,ConnectsVal):-
hf_ring_structure_connect_aux(StrX,AllSt,AllRings,RAs,Val),
NewPart is Part + Val,
hf_ring_structure_connect(Rest,AllSt,AllRings,RAs,NewPart,ConnectsVal).

hf_ring_structure_connect_aux(S1,AllSt,AllRings,RAs,V):-
hf_ring_structure_connect_point(S1,AllSt,AllRings,RAs,Point),
hf_ring_structure_connect_side(S1,AllSt,AllRings,RAs,Side),
hf_ring_structure_connect_flat(S1,AllSt,AllRings,RAs,Flat),
hf_ring_structure_connect_spoke(S1,AllSt,AllRings,RAs,Spoke),
V is Point + Side + Flat + Spoke.

/**************************************************************
Correction for structures joined at a point
**************************************************************/
hf_ring_structure_connect_point(S1,_All,AllRings,RAs,Point):-
        flatten(S1,FlatS1),
        delete_duplicates(FlatS1,RAsS1),
findall([A,B],(member(Atom,RAs),
       memberchk(Atom,RAsS1),
       member(N,AllRings),
       \+ memberchk(N,S1),
       memberchk(Atom,N),
       hf_which_ring(Atom,S1,A),
       findall(Q,(member(Q,AllRings),
  memberchk(Atom,Q),
  \+ memberchk(Q,S1)),ListOther),
       hf_which_ring(Atom,ListOther,B),
       intersects(A,B,[_])),List),
duplicate_list_sort(List,ListPoints),
ListPoints \= [],
!,
hf_rsc_point_aux2(ListPoints,0,Point).
hf_ring_structure_connect_point(_,_,_,_,0).

hf_rsc_point_aux2([],In,In).
hf_rsc_point_aux2([[A,B]|T],In,Out):-
hf_ring_structure2_point(A,B,Val),
NewIn is In + Val,
hf_rsc_point_aux2(T,NewIn,Out).

hf_ring_structure2_point(A,B,Val):-
        length(A,3),
        length(B,3),
        hf_ring(point3,Val).
hf_ring_structure2_point(_A,_B,Val):-
        hf_ring(point,Val).
hf_ring_structure2_point(_,_,0).

/*************************************************************
Correction for structures joined at a side
*************************************************************/
hf_ring_structure_connect_side(S1,All,AllRings,RAs,Side):-
        flatten(S1,FlatS1),
        delete_duplicates(FlatS1,RAsS1),

findall([Atom,A,S1,B,S2],(member(Atom,RAs),
       memberchk(Atom,RAsS1),
       member(N,AllRings),
       \+ memberchk(N,S1),
       memberchk(Atom,N),
       hf_which_ring(Atom,S1,A),
       findall(Q,(member(Q,AllRings),
  memberchk(Atom,Q),
  \+ memberchk(Q,S1)),ListOther),
       hf_which_ring(Atom,ListOther,B),
       hf_which_structure(B,All,[S2]),
       intersects(A,B,Intersects),
       length(Intersects,2),
       memberchk(Atom,Intersects)),List),

duplicate_list_sort(List,ListX),
ListX \= [],
!,
hf_rsc_side_aux(ListX,0,Side).
hf_ring_structure_connect_side(_,_,_,_,0).
```

```
hf_rsc_side_aux([],In,In).
hf_rsc_side_aux([[Atom,A,S1,B,S2]|T],In,Out):-
hf_ring_structure2_side(Atom,[A,S1,B,S2],Val),
NewIn is In + Val,
hf_rsc_side_aux(T,NewIn,Out).

/*********************************************************
1. Gets the correction for rings(A,B) joined by 2 atoms at a side

calls get_hf_ring_side w/ the two rings(A,B) gets the side correct

determines if the side atom has a branch, gets that correction

If A and B are in structures w/ mulitple rings that Atom is
a part of (ie bridge structure) get a correction for the
extra strain.
2. For rings that share a doublebond between them
already accounted for so get a 0.
*********************************************************/
hf_ring_structure2_side(Atom,[A,S1,B,S2],Val):-
bonded(Atom,1,Atom2),
memberchk(Atom2,A),
memberchk(Atom2,B),

        get_hf_ring_side(A,B,V1),

atom_specs(_,Atom,_,_,List),
        findall(Y,(member(Y,List),
                   \+ member(Y,A),
   \+ member(Y,B)), ListOut),
hf_ring_structure2_side_aux(ListOut,V2),

length(S1,LS1),
length(S2,LS2),
(LS1 > 1, LS2 >1 -> ring_vertices_2ring(bridge,V3)
          ; V3 is 0
),
Val is V1 + V2 + V3.
hf_ring_structure2_side(_,_,0).

hf_ring_structure2_side_aux([],0).
hf_ring_structure2_side_aux([LO],Val):-
substituent_type(_,LO,Type),
ring_vertices_2ring(Type,Mult),
ring_vertices_2ring(1,V),
Val is Mult*V.
hf_ring_structure2_side_aux([_],0).



/******************************************************************
        The first 3 rules deal with 3-membered rings sharing a side
        with another ring
        The 4th rule handles 5-5 molecules
        The 5th is a general rule that handles everything else
        ....see notebook
**********************************************************************/
get_hf_ring_side(H,R,Val) :-
        symmetric(H,R,AA,BB),
        length(AA,3),
        length(BB,L),
        L >= 3,
        L =< 5,
        hf_ring_side3(L,Val).
/***
        hf_ring_side3(a,A),
        hf_ring_side3(b,B),
        Val is A*log(L)+B.
***/
get_hf_ring_side(H,R,Val) :-
        symmetric(H,R,AA,BB),
        length(AA,3),
        length(BB,L),
        L >= 6,
        L =< 8,
        hf_ring_side3(L,Val).
get_hf_ring_side(H,R,Val) :-
        symmetric(H,R,AA,BB),
        length(AA,3),
        length(BB,L),
        L > 8,
        hf_ring_side3(8,Val).   %setting to the same as 8's value b/c don't
        %have any in set and it should level off somewhere
%near here
get_hf_ring_side(H,R,Val) :-
%correction for 5-5 (going to be cis naturally)
        symmetric(H,R,AA,BB),   %trans taken care of in chirality
        length(AA,5),
        length(BB,5),
        hf_ring_side(side5,Val).
```

```
get_hf_ring_side(H,R,Val) :-    %special correction for 6-6
        symmetric(H,R,A,B),
        length(A,6),
        length(B,6),
        hf_ring_side(side6,Val).
get_hf_ring_side(H,R,Val) :-    %all other rings
        symmetric(H,R,A,B),
        length(A,LA),
        length(B,LB),
        \+ memberchk(3,[LA,LB]),
        hf_ring_side(side,Val).

/***************************************************************
Correction for structures joined two other structure
on adjacent sides (eg flat structure)
***************************************************************/
hf_ring_structure_connect_flat(S1,_All,AllRings,_RAs,Flat):-
        flatten(S1,FlatS1),
        delete_duplicates(FlatS1,RAsS1),
        findall([A,B,C],(member(Atom1,RAsS1),
                         member(Atom2,RAsS1),
                         Atom1 \= Atom2,
                         bonded(Atom1,_,Atom2),
                         member(Atom3,RAsS1),
                         Atom1 \= Atom3,
                         Atom2 \= Atom3,
                         bonded(Atom1,_,Atom3),
                         hf_which_ring(Atom1,S1,A),
                         memberchk(Atom2,A),
                         memberchk(Atom3,A),
                         findall(Q,(member(Q,AllRings),
                                    memberchk(Atom1,Q),
                                    memberchk(Atom2,Q),
                                    \+ memberchk(Q,S1)),ListOther2),
                         hf_which_ring(Atom1,ListOther2,B),
                         findall(R,(member(R,AllRings),
                                    memberchk(Atom1,R),
                                    memberchk(Atom3,R),
                                    \+ memberchk(R,S1)),ListOther3),
                         hf_which_ring(Atom1,ListOther3,C)),List),
duplicate_list_sort(List,ListX),
ListX \= [],
!,
hf_rsc_fs_aux(ListX,flat,0,Flat).
hf_ring_structure_connect_flat(_,_,_,_,0).
```

```
hf_rsc_fs_aux([],_,In,In).
hf_rsc_fs_aux([_H|T],Type,In,Out):-
ring_vertices_3ring(Type,Val),
NewIn is In + Val,
hf_rsc_fs_aux(T,Type,NewIn,Out).


/***************************************************************
Correction for structures joined at one side to two other
structures (eg spoke)
***************************************************************/
hf_ring_structure_connect_spoke(S1,_All,AllRings,RAs,Spoke):-
        flatten(S1,FlatS1),
        delete_duplicates(FlatS1,RAsS1),
findall([A,B,C],(member(Atom1,RAs),
 member(Atom2,RAs),
 Atom1 \= Atom2,
         memberchk(Atom1,RAsS1),
         memberchk(Atom2,RAsS1),
 bonded(Atom1,_,Atom2),
         hf_which_ring(Atom1,S1,A),
 memberchk(Atom2,A),
         findall([Q,R],(member(Q,AllRings),
    member(R,AllRings),
    Q \= R,
    \+ intersects(Q,R,[_,_,_|_]),
    memberchk(Atom1,Q),
    memberchk(Atom2,Q),
    memberchk(Atom1,R),
    memberchk(Atom2,R),
    \+ memberchk(Q,S1),
            \+ memberchk(R,S1)),ListOther2),
 duplicate_list_sort(ListOther2,ListOther),
 hf_which_ring2(ListOther,[B,C]),
 intersects(A,B,[_,_]),
 intersects(B,C,[_,_]),
 intersects(A,C,[_,_])),List),
duplicate_list_sort(List,ListX),
ListX \= [],
!,
hf_rsc_fs_aux(ListX,spoke,0,Spoke).
hf_ring_structure_connect_spoke(_,_,_,_,0).
```

```
/*************************************************************
Asserts the structure into the database:

structure(Structure,SizeofRingsInStructure,TypeofStructure)
*************************************************************/
hf_ring_structure_assert([],_):- !.
hf_ring_structure_assert([H|T],Type):-
hf_ring_structure_assert_aux(H,ListLens),
assert(structure(H,ListLens,Type)),
hf_ring_structure_assert(T,Type).

hf_ring_structure_assert_aux([],[]).
hf_ring_structure_assert_aux([R1|Rest],[Len|ListLens]):-
length(R1,Len),
hf_ring_structure_assert_aux(Rest,ListLens).

/***********************************************************************
Utility routines for hf_ring2

1. hf_ring_find:  finds all of the structures of a specified type
2. hf_bridge_atom: finds the bridge atoms - those in all three rings
3. hf_which_ring: finds the all the rings a particular atom is in
  then sorts that list to find the shortest ring
4. hf_which_structure: finds the structure(s) of which a particular ring is
         a member
5. hf_atom_in3rings: finds all the atoms that are in 3 rings of a structure
6. hf_atom_in4rings: finds all the atoms that are in 4 rings of a structure
7. hf_which_ring2: finds the shortest X and Y rings in a list of [X,Y]s
8. hf_numes: finds the number of ethylenics in a ring
9. hf_ring_structure_filter:  finds only the independent structures
***********************************************************************/

hf_ring_find(Type,AllStructures):-
findall(X,structure(X,_,Type),AllStructures).

hf_bridge_atom(H,Atoms):-
H = [A,B,C],
findall(X,(member(X,A),
   member(X,B),
   member(X,C)),Atoms).

hf_which_ring(Atom,S1,ShortRingIn):-
findall(X,(member(X,S1),
   memberchk(Atom,X)),RingsAtomIn),
gensort(RingsAtomIn,longer,[ShortRingIn|_]).

hf_which_structure(Ring,AllStructures,Structure):-
findall(X,(member(X,AllStructures),
   memberchk(Ring,X)),LS),
delete_duplicates(LS,Structure).


hf_atom_in3rings(H,Atoms):-
flatten(H,FH),
delete_duplicates(FH,FH1),
findall(X,(member(X,FH1),
   member(Ring1,H),
   member(Ring2,H),
   Ring1 \= Ring2,
   member(Ring3,H),
   Ring1 \= Ring3,
   Ring2 \= Ring3,
   memberchk(X,Ring1),
   memberchk(X,Ring2),
   memberchk(X,Ring3)),Atoms1),
delete_duplicates(Atoms1,Atoms).

hf_atom_in4rings(H,Atoms):-
flatten(H,FH),
delete_duplicates(FH,FH1),
findall(X,(member(X,FH1),
   member(Ring1,H),
   member(Ring2,H),
   Ring1 \= Ring2,
   member(Ring3,H),
   Ring1 \= Ring3,
   Ring2 \= Ring3,
   member(Ring4,H),
   Ring1 \= Ring4,
   Ring2 \= Ring4,
   Ring3 \= Ring4,
   memberchk(X,Ring1),
   memberchk(X,Ring2),
   memberchk(X,Ring3),
   memberchk(X,Ring4)),Atoms1),
delete_duplicates(Atoms1,Atoms).

hf_which_ring2([],[]).
hf_which_ring2(L0,[ShortX,ShortY]):-
```

```
findall(X,(member([X,_],LO)),ListX),
delete_duplicates(ListX,List),
gensort(List,longer,[ShortX|_]),
findall(Y,(member([_,Y],LO)),ListY),
delete_duplicates(ListY,List2),
gensort(List2,longer,[ShortY|_]).

hf_numes(R1,Num5Es):-
findall(E,(reactophore_fact(E,_,Atoms,_,_),
   member(A,R1),
   member(B,R1),
   A \= B,
   memberchk(A,Atoms),
   memberchk(B,Atoms)),ListE),
delete_duplicates(ListE,List),
length(List,Num5Es).

hf_ring_structure_filter([],[]).
hf_ring_structure_filter(Structures,FilteredSts):-
gensort(Structures,shorter,[H|Structures2]),
hf_ring_structure_filter_aux(Structures2,[H],FilteredSts).

hf_ring_structure_filter_aux([],A,A).
hf_ring_structure_filter_aux([H|T],In,FilteredSts):-
length(H,LenH),
findall(X,(member(X,In),
   intersects(H,X,Y),
   length(Y,LenH)),[]),
hf_ring_structure_filter_aux(T,[H|In],FilteredSts).
hf_ring_structure_filter_aux([_|T],In,FilteredSts):-
hf_ring_structure_filter_aux(T,In,FilteredSts).




/**** Data Area *****************************************************/

% This is now in a file called 'hf_data.pro'
```

C.3 OTHER_AROM.PRO

```
ot_a:-
```

```
        batch_mode,
        findall(reactophore_fact(A,B,C,D,E),
         reactophore_fact(A,B,C,D,E),[List]),
        oa_aux(List,HF),
        assert(value_calculated(HF)).

oa(Smiles,Hf) :-
        execute(prepare_for_property(nul,Smiles),_ResTop),
        findall(reactophore_fact(A,B,C,D,E),
           reactophore_fact(A,B,C,D,E),[List]),
        oa_aux(List,Hf).

/*********************************************************************
        Gets the basic correction for aromatic compounds
*********************************************************************/
oa_aux(reactophore_fact(R1,_,_,_,_),HF) :-
        reac_type(R1,_,aromatic),
        oa_correct(aromatic,R1,HF).

/*********************************************************************
        Correction to Hf for each of the reactophore types
R1 is the aromatic reactophore
List  is the list of atoms in the reactophore...except
for the buried atoms
List1 is the list of bridge atoms (joins two rings on
the edge of ring)
List2 is the list of buried atoms (atoms not on outer
edge of ring)
Correction is the (#of each type)*correction + bent + resonance

need to add something that can handle nitrogens too
*********************************************************************/
oa_correct(aromatic,R1,Correction) :-
reactophore_fact(R1,_,List,_,_),
oa_contrib(part,Val),

bridge_atoms(R1,List1),
length(List1,NumAtoms1),
brg_contrib(part,Val1),

buried_atoms(R1,List2),
length(List2,NumAtoms2),
buried_contrib(part,Val2),

append(List2,List,ListTotal),
oa_extended_res(ListTotal,NumPhen,ResCorrect),

oa_bent(ListTotal,List1,List2,NumPhen,BentCorrect),

oa_helical(List1,HelicalCorrect),

length(ListTotal,NumAtoms),

Correction is (NumAtoms * Val) + (NumAtoms1 * Val1) + (NumAtoms2 * Val2) +
      BentCorrect + ResCorrect + HelicalCorrect,

        !.
oa_correct(_,_,0).

/*************************************************************************
        Correction for the strain introduce when the molecule is helical,
eg benzo[c]phenanthrene.

Should be ~33kJ, but this may be an experimental error, so not
training.

for 1 set gets 33, for 2 sets gets 66, etc

gets all phenyl_rings
finds A,B,C,D & the corresponding BridgeAtoms of these PRs....
where AB1 is in BrA & BrB and BC1 is in BrB & BrC and AB1 is bonded
to BC1  and CD1 is in BrC and BrD and CD1 is bonded to BC1

*************************************************************************/
oa_helical(BrAtoms,Correct) :-
findall(X,phenyl_ring(X),PRs1),
fix_phenyl_rings(PRs1,PRs1,PRs),
findall(Set,(member(A,PRs),
     intersects(A,BrAtoms,BrA),
     member(B,PRs),
     B \= A,
     intersects(B,BrAtoms,BrB),
         member(C,PRs),
     C \= B,
     C \= A,
     intersects(C,BrAtoms,BrC),
     member(D,PRs),
     D \= C,
     D \= B,
     D \= A,
     intersects(D,BrAtoms,BrD),
     member(AB1,BrA),
     memberchk(AB1,BrB),
     member(BC1,BrC),
     memberchk(BC1,BrB),
     bonded(AB1,_,BC1),
        member(CD1,BrC),
     memberchk(CD1,BrD),
     bonded(CD1,_,BC1),
     Set=[A,B,C,D]
    ),List),
duplicate_list_sort(List,RealSets),
length(RealSets,LRS),
helical(part,Val),
Correct is LRS * Val.
oa_helical(_,0).


/************************************************************************
        Correction for the number of phenyls in a compound b/c as they
increase the amount of resonance will increase b/c more charge gets
distributed around the reactophore
************************************************************************/
oa_extended_res(Atoms,Len,Value) :-
findall(X,(phenyl_ring(X),
                   intersects(X,Atoms,Y),
   difference(X,Y,[])),List),
length(List,Len),
Len > 1,
!,
res_contrib(part,Val),
Value is Val*Len.
oa_extended_res(_,1,0).

/************************************************************************
Deals with reactophores where it is "bent"...more resonance occurs
on the outside of the bend rather than the inside...so this predicate
takes this into account and normalizes the correction for the number of
phyenls in the compound

ListPR is the list of phyenl rings
FixedPR is the list of phyenl rings corrected so all the atoms show up
FilteredPR is the list of phyenl rings where PRs that have >3 buried
atoms are removed
get_3_combos returns the number of bends in a compound
```

```
LenP is the number of phyenl rings in the reactophore

This finds the number of bends and normalizes it vs the number of
    phenyls (LenP) and multiplies that by the correction val.
***************************************************************************/
oa_bent(AllAtoms,BridgeAtoms,BuriedAtoms,LenP,Value) :-
findall(PR,(phenyl_ring(PR),
                    intersects(PR,AllAtoms,Y),
    difference(PR,Y,[])),ListPR),
fix_phenyl_rings(ListPR,ListPR,FixedPR),
filter_pr(FixedPR,BuriedAtoms,FilteredPR),
get_3_combos(FilteredPR,BuriedAtoms,BridgeAtoms,AllAtoms,NumBends),
Mult is NumBends / LenP,
bent_contrib(part,Val),
Value is Val * Mult.


/***************************************************************************
removes PRs that have more than 3 buried atoms
***************************************************************************/
filter_pr([],_,[]).
filter_pr([H|T],BuriedAtoms,[H|Rest]):-
intersects(H,BuriedAtoms,Overlap),
length(Overlap,LenOl),
LenOl < 4,
!,
filter_pr(T,BuriedAtoms,Rest).
filter_pr([_|T],BuriedAtoms,Rest):-
filter_pr(T,BuriedAtoms,Rest).

/***************************************************************************
Finds the number of bends in a reactophore

Finds atom(s) that are part of either buried or bridge atoms
and bonded to each other,  then finds PRs that either
have both X&Y in it (Z), and X or Y (ZZ and ZZZ),
the IF statement is to calculate whether the bend is
   inner or outer--if it's an outer it gets full correction
an inner it gets only 1/2 of a full correction.
   the inner bend occurs when the 2PRs (ZZ and ZZZ) are
connected ONLY!! by "c" atoms in ZZZ
   the outer bend occurs when the 2PRs (ZZ and ZZZ) are
connected both by "c" and "cH" atoms in ZZZ
```

```
Works b/c PR does not contain the buried atom PRs
**************************************************************************/
get_3_combos(PR,BuA,BrA,AA,NumBends):-
append(BuA,BrA,ABs),
difference(AA,ABs,RegAtoms),
findall(XY,(member(X,ABs),
     member(Y,ABs),
     Y \= X,
     bonded(X,_,Y),
     member(Z,PR),
     intersects([X,Y],Z,[_,_]),
     member(ZZ,PR),
     ZZ \= Z,
     intersects([X,Y],ZZ,[X]),
     member(ZZZ,PR),
     ZZZ \= ZZ,
     ZZZ \= Z,
     intersects([X,Y],ZZZ,[Y]),
     (intersects(Z,RegAtoms,List),
      member(X1,List),
      member(X2,List),
      bonded(X1,_,X2) -> XY is 1
       ; XY is 0.5
     )),BendsX),
sumlist(BendsX,Num),
NumBends is Num / 2.

/**** Data Area ************************************************************/

% This is now in a file called 'hf_data.pro'
```

C.4 HF_chiral.pro

```
/*********************************************************************
Gets the chiral correction for molecules
_sets - breaks ListCA into types of CAs
_aux1 - for List1 types
_aux2 - for List2 types
_ring_sep - breaks up ringCAs into 3 types
_aux_single - for CAs in single rings
_aux_bridge - for CAs in bridge rings
_aux_sideshare - for CAs in sideshare rings
```

```
*****************************************************************/
hf_chiral(AllRings,Chiral):-
findall(CA,chiral_atom(CA,_),ListCA),
ListCA \= [],
hf_chiral_sets(ListCA,AllRings,[List1,List2,CASets,_]),
hf_chiral_aux1(List1,AllRings,0,Chiral1),
hf_chiral_aux2(List2,AllRings,0,Chiral2),
hf_chiral_ring_sep(CASets,[Single,Bridge,SideShare]),
hf_chiral_aux_single(Single,AllRings,0,CSingle),
hf_chiral_aux_bridge(Bridge,AllRings,0,CBridge),
hf_chiral_aux_sideshare(SideShare,AllRings,0,CSideShare),
Chiral is Chiral1 + Chiral2 + CSingle + CBridge + CSideShare.
hf_chiral(_,0).

/*****************************************************************
separates the CASets into the 3 type of ring chirals
*****************************************************************/
hf_chiral_ring_sep([],[[],[],[]]). %no CAs
hf_chiral_ring_sep([Set-Ring],[[Set-Ring],[],[]]):- !. %single ring CAs
hf_chiral_ring_sep(CASets,[Single,Bridge,SideShare]):-  %mulitple CAs
hf_chiral_remove_bridge(CASets,Bridge1,RestnoBridge),
hf_chiral_sort_sets(Bridge1,Bridge),
hf_chiral_remove_sideshare(RestnoBridge,SideShare1,RestnoSide),
hf_chiral_sort_sets(SideShare1,SideShare),
hf_chiral_ring_sep_rest(RestnoSide,[],Single).

/*****************************************************************
sorts the Sets into individual groups [[Bridge1],[Bridge2],...]

_aux    groups all those rings that intersect in 2 places together
put each ring in the donelist and then on the recall
checks the done list to make sure that the ring working on
intersects this donelist

then recalling
*****************************************************************/
hf_chiral_sort_sets([],[]).
hf_chiral_sort_sets(SetsIn,[Same|Sets]):-
hf_chiral_sort_sets_aux(SetsIn,[],[],Same),
difference(SetsIn,Same,Rest),
hf_chiral_sort_sets(Rest,Sets).

%%%
hf_chiral_sort_sets_aux(Sets,DoneList,Same,Out):-
member(_-Ring1,Sets),
\+ memberchk(Ring1,DoneList),
(DoneList = [] -> true
; findall(RR,(member(RR,DoneList),
          intersects(RR,Ring1,[_,_|_])),[_|_])
),
findall(AB-R,(member(AB-R,Sets),
      \+ memberchk(R,DoneList),
      intersects(R,Ring1,[_,_|_])),List),
List \= [],
!,
append(List,Same,NewSame),
hf_chiral_sort_sets_aux(Sets,[Ring1|DoneList],NewSame,Out).
hf_chiral_sort_sets_aux(_,_,InSame,Same):-
delete_duplicates(InSame,Same).

/*********************************************************************
gets the sets which are part of the bridge section of molecule
and the remaining sets
*********************************************************************/
hf_chiral_remove_bridge(CASets,BridgeSets,RestSets):-
findall(Rings,structure(Rings,_,bridge),ListRings1),
lol3_to_lol2(ListRings1,[],ListRings),
ListRings \= [],
findall(Set-Ring,(member(Ring,ListRings),
         member(Set-Ring,CASets)),PartBridgeSet1),
hf_chiral_remove_bridge_aux(PartBridgeSet1,CASets,[],PBS2),
delete_duplicates(PBS2,PartBridgeSet2),
append(PartBridgeSet1,PartBridgeSet2,BridgeSets),
difference(CASets,BridgeSets,RestSets).
hf_chiral_remove_bridge(CASets,[],CASets).

/*********************************************************************
gets the sets making up the bridge section itself
*********************************************************************/
hf_chiral_remove_bridge_aux([],_,Out,Out).
hf_chiral_remove_bridge_aux([_-Ring|T],CASets,In,Part):-
findall(Set-ORing,(member(Set-ORing,CASets),
   intersects(Ring,ORing,[_,_])),List),
List \= [],
append(List,In,NewIn),
hf_chiral_remove_bridge_aux(T,CASets,NewIn,Part).
hf_chiral_remove_bridge_aux([_|T],CASets,In,Part):-
hf_chiral_remove_bridge_aux(T,CASets,In,Part).

/*********************************************************************
```

```
gets the sets part of sideshare section of molecule and the remaining sets
******************************************************************/
hf_chiral_remove_sideshare(RestnoBridge,SideSets,RestnoSide):-
hf_chiral_set_rings(RestnoBridge,Rings),
findall(Set-Ring,(member(Set-Ring,RestnoBridge),
  member(Ring,Rings),
  member(ORing,Rings),
  ORing \= Ring,
  intersects(Ring,ORing,[_,_])),TempSideSets),
delete_duplicates(TempSideSets,SideSets),
difference(RestnoBridge,SideSets,RestnoSide).

/******************************************************************
seperates the remaining sets into Single
(could be expanded if necessary)
******************************************************************/
hf_chiral_ring_sep_rest(Single,_,Single).

/******************************************************************
hf_chiral_sets(ListCA,AllRings,[List1,List2,CASets,Rest])

Finds the sets of CAs [List1,List2,RingSets,Rest]

List1 are CAs bonded together where X is not in a ring [[SetL1],...]
List2 are CAs seperated by 2 bonds where X is not in a ring [[SetL2],...]
RingSets are the lists of CAs in rings  [[Set1]-Ring1,[Set2]-Ring2,...]
Rest are the remaining CAs (maybe need rule for?)

chiral_check returns those CAs that aren't used in L1,L2,orRing
******************************************************************/
hf_chiral_sets(ListCA,AllRings,[L1Sets,L2Sets,RingSets,Rest]):-
flatten(AllRings,RAs1),
delete_duplicates(RAs1,RAs),
hf_chiral_sets_list1(ListCA,ListCA,RAs,L1Sets),
hf_chiral_sets_list2(ListCA,ListCA,RAs,L2Sets),
hf_chiral_sets_rings(ListCA,AllRings,RingCAs),
hf_chiral_sets_aux2(AllRings,RingCAs,RingSets),
hf_chiral_check(ListCA,L1Sets,L2Sets,RingCAs,Rest).

%%%%
hf_chiral_check(ListCA,L1Sets,L2Sets,RingCAs,Rest):-
append(L2Sets,L1Sets,AllSets1),
append(RingCAs,AllSets1,AllSets),
flatten(AllSets,FSets),
delete_duplicates(FSets,ListCA1),
difference(ListCA,ListCA1,Rest).

%%%
hf_chiral_sets_aux2(_,[],[]). %no RingCAs
hf_chiral_sets_aux2([],_,[]).
hf_chiral_sets_aux2([H|T],RingCAs,[Set-H|RingSets]):-
findall(X,(member(X,RingCAs),
   memberchk(X,H)),Set),
Set \= [],
hf_chiral_sets_aux2(T,RingCAs,RingSets).
hf_chiral_sets_aux2([_|T],RingCAs,RingSets):-
hf_chiral_sets_aux2(T,RingCAs,RingSets).

%%%
hf_chiral_sets_rings([],_,[]).
hf_chiral_sets_rings([H|T],AllRings,[H|RingCAs]):-
findall(Ring,(member(Ring,AllRings),
      memberchk(H,Ring)),[_|_]),
hf_chiral_sets_rings(T,AllRings,RingCAs).
hf_chiral_sets_rings([_|T],AllRings,RingCAs):-
hf_chiral_sets_rings(T,AllRings,RingCAs).

%%%
hf_chiral_sets_list1([],_,_,[]).
hf_chiral_sets_list1([X|T],ListCA,RAs,[X|List1CAs]):-
\+ memberchk(X,RAs),
bonded(X,1,Y),
memberchk(Y,ListCA),
hf_chiral_sets_list1(T,ListCA,RAs,List1CAs).
hf_chiral_sets_list1([_|T],ListCA,RAs,List1CAs):-
hf_chiral_sets_list1(T,ListCA,RAs,List1CAs).

%%%
hf_chiral_sets_list2([],_,_,[]).
hf_chiral_sets_list2([X|T],ListCA,RAs,[X|List1CAs]):-
\+ memberchk(X,RAs),
bonded(X,1,Z),
\+ memberchk(Z,RAs),
bonded(Z,1,Y),
Y \= X,
memberchk(Y,ListCA),
hf_chiral_sets_list2(T,ListCA,RAs,List1CAs).
hf_chiral_sets_list2([_|T],ListCA,RAs,List1CAs):-
hf_chiral_sets_list2(T,ListCA,RAs,List1CAs).
```

```
/******************************************************************
Gets the chiral correction for each set of chiral pairs
Finds all the chiral_pairs that have already been
    corrected for
Gets the list of rings with these chiral atoms in them
Passes this info into the _aux routine where the
    real work occurs

_aux1 for chiral atoms bonded to each other and one not in a ring
get a 0 b/c chain bends out of way, so can't distinguish
difference in UP,UP and UP,DOWN

_aux2 for chiral atoms w/ an atom between them and one not in a ring
same as _aux1, this could possibly be worth ~6kJ per interaction
but the measurements cover this up b/c in some cases the error is
+/-6kJ, so leaving at 0
_aux_single - loops over all the sets:
1 atom in set: get 0
mutiple atoms in 1 set:
assert the size of the substituents (chiral_size)
rotate ring so largest substituent is first
get the senses of the set
get the locations (AxInt)
get the neighboring interactions (ChiInt)
get the other interactions (i.e. oh and methyl interacting)
call rule1 - get stereoisomer correction based on above interactions

_aux_bridge
gets the bridge sets and correction
_aux_side
gets the sideshare sets and correction
******************************************************************/
hf_chiral_aux1(_,_,Out,Out).

%%%%
hf_chiral_aux2(_,_,Out,Out).

/******************************************************************
takes care of compounds that are just regular CAs
******************************************************************/
hf_chiral_aux_single([],_,Rest,Rest).
hf_chiral_aux_single([H|T],AllRings,Part,Rest):-
hf_chiral_aux3([H],AllRings,0,Val),
NewPart is Part + Val,
hf_chiral_aux_single(T,AllRings,NewPart,Rest).
```

```
/*****************************************************************
takes care of compounds that have CAs part of a bridge
*****************************************************************/
hf_chiral_aux_bridge([],_,CBridge,CBridge).
hf_chiral_aux_bridge([H|BSets],AllRings,Part,CBridge):-
hf_chiral_bridge_aux(H,AllRings,Val),
NewPart is Part + Val,
hf_chiral_aux_bridge(BSets,AllRings,NewPart,CBridge).

/*****************************************************************
takes care of compounds that have CAs sharing a side
*****************************************************************/
hf_chiral_aux_sideshare([],_,Out,Out).
hf_chiral_aux_sideshare([H|SideSets],AllRings,Part,CSideShare):-
hf_chiral_sideshare_aux(H,AllRings,Val),
NewPart is Part + Val,
hf_chiral_aux_sideshare(SideSets,AllRings,NewPart,CSideShare).

/*****************************************************************
Single atom work done here
_set_data_single: sets the ring up w/ largest sub first
_sense: gets the senses of newly rotated ring
_get_interactions: gets all interactions in ring
_rule1: gets the Chiral value from interactions
*****************************************************************/
hf_chiral_aux3([[_]-_],_,Out,Out):- !. %one atom in set
hf_chiral_aux3([Set-Ring],_,_,Chiral):-%multiple atoms in one set
hf_chiral_set_data_single(Set,Ring,NewRing),
hf_chiral_sense(NewRing,Senses),
length(NewRing,RLen),
hf_chiral_get_interactions(RLen,Set-NewRing,Senses,ChiInt,AxInt,OHME,NO2),
hf_chiral_rule1(RLen,Set-NewRing,ChiInt,AxInt,OHME,NO2,Chiral).

/*****************************************************************
have to determine the "relative" sense of the side sets
need the size of the rings that are sharing a side
*****************************************************************/
hf_chiral_sideshare_aux(SideSets,AllRings,SideShare):-
hf_chiral_set_rings(SideSets,SetRings),
hf_chiral_set_data_sideshare(SideSets,SetRings),
/**
hf_chiral_set_senses(SideSets,SideRings,SideSets),
**/
hf_chiral_sideshare_aux_aux(SideSets,AllRings,[],[],SideShare).
```

```
%(why allrings and not SetRings?)

%%%
hf_chiral_sideshare_aux_aux([],AllRings,AllSets,Ring3Sets,Val):-
delete_duplicates(AllSets,AllSets1),
length(AllSets1,LAS),
LAS > 1,
hf_chiral_sideshare_aux_aux2(AllSets1,AllRings,[],SSCAs,OtherCAs),
hf_chiral_sideshare_aux_aux3(SSCAs,SSCAs,OtherCAs,AllRings,0,V1),
hf_chiral_sideshare_aux_aux4(Ring3Sets,AllRings,0,V2),
Val is V1 + V2.
hf_chiral_sideshare_aux_aux([],_,_,_,0).
hf_chiral_sideshare_aux_aux([Set-Ring|Rest],AllRings,In,InK,Val):-
hf_chiral_sideshare_2ring(Set-Ring,AllRings,InK,Keep),
append(Set,In,NewSet),
hf_chiral_sideshare_aux_aux(Rest,AllRings,NewSet,Keep,Val).

/*************************************************
find if a set is sharing a side w/ at least 2 rings
**************************************************/
hf_chiral_sideshare_aux_aux4([],_,V2,V2).
hf_chiral_sideshare_aux_aux4([Set-Ring|T],AllRings,Part,V2):-
findall([A1,A2,ORing],(member(A1,Set),
         member(A2,Set),
         bonded(A1,_,A2),
         member(ORing,AllRings),
         ORing \= Ring,
         memberchk(A1,ORing),
         memberchk(A2,ORing)),SS1),
duplicate_list_sort(SS1,SS),
hf_csaa4_aux(SS,Senses),
hf_csaa4_aux2(SS,[R1,R2]),

length(Ring,RS),
hf_chiral_rule4(RS,R1,R2,Senses,Val),

NewPart is Part + Val,
hf_chiral_sideshare_aux_aux4(T,AllRings,NewPart,V2).
hf_chiral_sideshare_aux_aux4([_|T],AllRings,Part,V2):-
hf_chiral_sideshare_aux_aux4(T,AllRings,Part,V2).


%%%%
hf_csaa4_aux([],[]).
hf_csaa4_aux([[A,B,_]|T],[S1|Senses]):-
```

```
hf_chiral_sense([A,B],S1),
hf_csaa4_aux(T,Senses).

%%%
hf_csaa4_aux2([],[]).
hf_csaa4_aux2([[_,_,R]|T],[RLen|Rings]):-
length(R,RLen),
hf_csaa4_aux2(T,Rings).

/*********************************************
find if a set is sharing a side w/ at least 2 rings
**********************************************/
hf_chiral_sideshare_2ring(Set-Ring,AllRings,In,Keep):-
length(Set,LSet),
LSet > 3,
delete(AllRings,Ring,RestRings),
findall(X,(member(X,RestRings),
   member(Y,Set),
   member(Z,Set),
   Y \= Z,
   memberchk(Y,X),
   memberchk(Z,X)),List),
delete_duplicates(List,NewList),
length(NewList,NumNewList),
NumNewList > 1,
append([Set-Ring],In,Keep).
hf_chiral_sideshare_2ring(_,_,In,In).

/************************************************************
For when two rings are sharing a side

hf_chiral_sideshare_aux_aux3(SSCAS,SSCAs,OtherCAs,AllRings,Part,Val)
1. ending rule - call _aux_aux3_aux to get correction for
   otherCAs in molecule (like off a single ring)
2. gets the two CAs and two rings making up the shared side
   checks if either R1 or R2 have DBs in them 1 or 2 atoms from CA
   gets the value based on the size of the R1 and R2
   remove the OSSCA from the list and recall

************************************************************/
hf_chiral_sideshare_aux_aux3([],SSCAs,OtherCAs,AllRings,Part1,Val):-
hf_chiral_sideshare_aux_aux3_aux(OtherCAs,SSCAs,AllRings,0,Part2),
Val is Part1 + Part2.
hf_chiral_sideshare_aux_aux3([H|T],SSCAs,OtherCAs,AllRings,Part,Val):-
findall(X,(member(X,AllRings),
   memberchk(H,X)),[R1,R2]),
member(OSSCA,T),
memberchk(OSSCA,R1),
memberchk(OSSCA,R2),
bonded(H,_,OSSCA),
length(R1,SizeR1),
length(R2,SizeR2),
(SizeR1 = SizeR2 -> SizeR1 = Short,
    SizeR2 = Long
  ; sort([SizeR1,SizeR2],[Short,Long])
),
hf_chiral_sense([H,OSSCA],Senses),

hf_chiral_sideshare_dbcheck([R1,R2],H,OSSCA,[],DBCheck),
length(DBCheck,NumDBRings),

hf_chiral_rule3(Short,Long,1,NumDBRings,Senses,Chiral),

NewPart is Part + Chiral,
delete(T,OSSCA,NewT),
hf_chiral_sideshare_aux_aux3(NewT,SSCAs,OtherCAs,AllRings,NewPart,Val).

/*****************************************************************
Get's the correction for the OtherCAs in the molecule
*****************************************************************/
hf_chiral_sideshare_aux_aux3_aux([],_,_,Part2,Part2).
hf_chiral_sideshare_aux_aux3_aux([H|T],SSCAs,AllRings,Part,Part2):-
findall([H,X],(chiral_atom(X,_),
       bonded(X,_,H)),[HSet]),
hf_chiral_sets(AllRings,HSet,[],[_,_,CASets,_]),
hf_chiral_remove_sets2(CASets,SSCAs,H,NewSets),
hf_chiral_aux3(NewSets,AllRings,0,Val),
NewPart is Part + Val,
hf_chiral_sideshare_aux_aux3_aux(T,SSCAs,AllRings,NewPart,Part2).
hf_chiral_sideshare_aux_aux3_aux([_|T],SSCAs,AllRings,Part,Part2):-
hf_chiral_sideshare_aux_aux3_aux(T,SSCAs,AllRings,Part,Part2).

%%%%
hf_chiral_remove_sets2([],_,_,[]).
hf_chiral_remove_sets2([CAs-Ring|T],SSCAs,OCA,[CAs-Ring|NewSets]):-
memberchk(OCA,CAs),
hf_chiral_remove_sets2(T,SSCAs,OCA,NewSets).
hf_chiral_remove_sets2([_|T],SSCAs,OCA,NewSets):-
hf_chiral_remove_sets2(T,SSCAs,OCA,NewSets).
```

```
/*****************************************************************
checks if R1 or R2 has a db 1 or 2 atoms from CA
*****************************************************************/
hf_chiral_sideshare_dbcheck([],_,_,Out,Out).
hf_chiral_sideshare_dbcheck([H|T],CA1,CA2,In,DBCheck):-
(hf_csd_aux(CA1,H) ; hf_csd_aux(CA2,H)),
hf_chiral_sideshare_dbcheck(T,CA1,CA2,[H|In],DBCheck).
hf_chiral_sideshare_dbcheck([_|T],CA1,CA2,In,DBCheck):-
hf_chiral_sideshare_dbcheck(T,CA1,CA2,In,DBCheck).

%%%%
hf_csd_aux(CA,Ring):-
member(A,Ring),
A \= CA,
bonded(CA,_,A),
\+ atom_specs(none,A,_,_,_).
hf_csd_aux(CA,Ring):-
member(A1,Ring),
A1 \= CA,
member(A2,Ring),
A2 \= CA,
A1 \= A2,
bonded(A1,_,A2),
\+ atom_specs(none,A2,_,_,_).

/*****************************************************************
Sorts the CAs into Chiral SideShare Atoms and Other Chiral Atoms
*****************************************************************/
hf_chiral_sideshare_aux_aux2([],_,Out,Out,[]) :- !.
hf_chiral_sideshare_aux_aux2([H|T],AllRings,InCSS,Out,OtherCAs):-
findall(X,(member(X,AllRings),
   memberchk(H,X)),[_,_]),
append(T,InCSS,Test),
member(A,Test),
bonded(H,_,A),
hf_chiral_sideshare_aux_aux2(T,AllRings,[H|InCSS],Out,OtherCAs).
hf_chiral_sideshare_aux_aux2([H|T],AllRings,InCSS,Out,[H|OtherCAs]):-
hf_chiral_sideshare_aux_aux2(T,AllRings,InCSS,Out,OtherCAs).

/*****************************************************************
Where the real work for bridge compounds is done
- finds if CBA is bonded to another CA
- check if this CA is bonded to another CBA
-if so then find out if
- up,up,up (chiral*2)
- up,up,down (chiral*0)
- down,up,down (chiral*4)
-if not - check if CA is in another ring and the size of that ring
-if ring is same direction as bridge -> 0
-if ring is opposite direction as bridge -> 1
-if that ring is <5 then get (chiral*~3)
-if ring >4 then get (chiral*1)
- if not in ring
-get 0 if direction is X,X
-get chiral if direction is X,Y
*******************************************************************/
hf_chiral_bridge_aux(BridgeSets,AllRings,Bridge):-
hf_chiral_bridge_aux_remove(BridgeSets,NewBridgeSets),
hf_chiral_set_rings(NewBridgeSets,SetRings),
hf_chiral_set_data_bridge(NewBridgeSets,SetRings),
hf_chiral_bridge_aux_aux(NewBridgeSets,AllRings,[],Bridge).
%hf_chiral_bridge_aux(_,_,0). %should never come here!

/*******************************************************************
_aux_aux2 Sorts the Sets into ChiralBridgeAtoms and OtherCAs
_aux_aux3 gets the correction
*******************************************************************/
hf_chiral_bridge_aux_aux([],AllRings,AllSets,Bridge):-
delete_duplicates(AllSets,AllSets1),
hf_chiral_bridge_aux_aux2(AllSets1,AllRings,[],CBAs,[],OtherCAs),
hf_chiral_bridge_aux_aux3(OtherCAs,CBAs,AllRings,0,Bridge).
hf_chiral_bridge_aux_aux([Set-_|Rest],AllRings,In,Bridge):-
append(Set,In,NewSet),
hf_chiral_bridge_aux_aux(Rest,AllRings,NewSet,Bridge).

/*******************************************************************
ending rule
for CA joining two CBAs    (two bridge sections joined by a CA)
for CA attached to one CBA
goes to _aux_aux3
for CA in bridge cmpd but not above case
should get the other rule1 effects (have to write)
maybe should recall w/ top level hf_chiral_aux3
*******************************************************************/
hf_chiral_bridge_aux_aux3([],_,_,Bridge,Bridge).
hf_chiral_bridge_aux_aux3([H|T],CBAs,AllRings,Part,Bridge):-
findall(X,(member(X,CBAs),
   bonded(H,_,X)),[A1,A2]),
hf_chiral_sense([A1,H,A2],Senses),
hf_chiral_rule2(bridgejoin,_,_,Senses,Val),
NewPart is Part + Val,
hf_chiral_bridge_aux_aux3(T,CBAs,AllRings,NewPart,Bridge).
hf_chiral_bridge_aux_aux3([H|T],CBAs,AllRings,Part,Bridge):-
findall(X,(member(X,CBAs),
   bonded(H,_,X)),[X]),
hf_chiral_bridge_aux_aux3_aux([X,H],AllRings,Val),
NewPart is Part + Val,
hf_chiral_bridge_aux_aux3(T,CBAs,AllRings,NewPart,Bridge).
hf_chiral_bridge_aux_aux3([H|T],CBAs,AllRings,Part,Bridge):-
findall(X,(member(X,CBAs),
   bonded(H,_,X)),[]),

%%% SHOULD GET THE OTHER "rule1" EFFECTS

hf_chiral_bridge_aux_aux3(T,CBAs,AllRings,Part,Bridge).

/****************************************************************
calls rule2(bridgering,...
if CA is part of a ring that does not make up the bridge cmpd
calls rule2(bridgechain,...
if CA is part of a chain
****************************************************************/
hf_chiral_bridge_aux_aux3_aux([CBA,CA],AllRings,Val):-
findall(Ring,(member(Ring,AllRings),
      \+ memberchk(CBA,Ring),
      memberchk(CA,Ring)),[Ring]),
length(Ring,Size),
(Size > 4 -> SubMult is 1
   ; SubMult is 2.5
),
hf_chiral_sense([CBA,CA],Senses),
hf_chiral_rule2(bridgering,Ring,SubMult,Senses,Val).
hf_chiral_bridge_aux_aux3_aux([CBA,CA],_,Val):-
hf_chiral_sense([CBA,CA],[CBAS,CAS]),
chiral_atom_info(CA,_,CAS,B1,_,_),
hf_chiral_size(CA-B1,SubMult),
hf_chiral_rule2(bridgechain,_,SubMult,[CBAS,CAS],Val).

/****************************************************************
Sorts the CAs into Chiral Bridge Atoms and Other Chiral Atoms
****************************************************************/
hf_chiral_bridge_aux_aux2([],_,CBAs,CBAs,Others,Others).
hf_chiral_bridge_aux_aux2([H|T],AllRings,InCBA,CBAs,InOther,OtherCAs):-
findall(X,(structure(Rings,_,bridge),
   member(X,Rings),
   memberchk(H,X)),[_,_,_]),
hf_chiral_bridge_aux_aux2(T,AllRings,[H|InCBA],CBAs,InOther,OtherCAs).
hf_chiral_bridge_aux_aux2([H|T],AllRings,InCBA,CBAs,InOther,OtherCAs):-
hf_chiral_bridge_aux_aux2(T,AllRings,InCBA,CBAs,[H|InOther],OtherCAs).

/*******************************************************************
removes the two short rings in a bridge
- fine for 5,5,6 have to worry about if ever get other kinds!!
*******************************************************************/
hf_chiral_bridge_aux_remove(BridgeSets,NewBridgeSets):-
findall([Set1-S1,Set2-S2],(structure(Rings,_,bridge),
             longest(Rings,Base),
      delete(Rings,Base,[S1,S2]),
             memberchk(Set1-S1,BridgeSets),
             memberchk(Set2-S2,BridgeSets)),Trash1),
lol3_to_lol2(Trash1,[],Trash),
hf_chiral_remove_sets(Trash,BridgeSets,NBS1),
delete_duplicates(NBS1,NewBridgeSets).

/*******************************************************************
 Get's the rings in a set
*******************************************************************/
hf_chiral_set_rings([],[]).
hf_chiral_set_rings([_-Ring|T],[Ring|Rings]):-
hf_chiral_set_rings(T,Rings).

/*******************************************************************
hf_chiral_get_interactions(RLen,Set-NewRing,Senses,ChiInt,AxInt,SpecInt)
gets the interactions in a ring:
0 for a chain
rules 3 to 6 for rings composed of 3-6 atoms

_interactions_aux (finds CAs next to each other, pointing same direction)
_interactions_aux2 (finds CAs in "Axial" location)
_interactions_other (for CAs w/ other interactions)
_interactions_aux3 (NEED TO DOCUMENT THIS)
*******************************************************************/
hf_chiral_get_interactions(0,Atoms,Senses,ChiInt,_,0,1):-
hf_chiral_get_interactions_aux(0,Atoms,Senses,[],_,ChiInt).
hf_chiral_get_interactions(3,Set-Ring,Senses,ChiInt,[],OHME,NO2):-
hf_chiral_get_interactions_aux(3,Ring,Senses,[],_,ChiInt),
hf_chiral_get_interactions_other(Set,Ring,Senses,ChiInt,_,OHME,NO2).
hf_chiral_get_interactions(4,Set-Ring,Senses,ChiInt,AxInt,OHME,NO2):-
hf_chiral_get_interactions_aux(4,Ring,Senses,[],S0,ChiInt),
hf_chiral_get_interactions_aux2(even,Ring,Senses,S0,[],AxInt),
hf_chiral_get_interactions_other(Set,Ring,Senses,ChiInt,_,OHME,NO2).
hf_chiral_get_interactions(5,Set-Ring,Senses,NewChiInt,[],OHME,NO2):-
hf_chiral_get_interactions_aux(5,Ring,Senses,[],_,ChiInt),
hf_chiral_get_interactions_other(Set,Ring,Senses,ChiInt,NewChiInt,OHME,NO2).
hf_chiral_get_interactions(6,Set-Ring,Senses,ChiInt,AxInt,OHME,NO2):-
hf_chiral_get_interactions_aux(6,Ring,Senses,[],S0,ChiInt),
hf_chiral_get_interactions_aux2(even,Ring,Senses,S0,[],AxInt),
hf_chiral_get_interactions_other(Set,Ring,Senses,ChiInt,_,OHME,NO2).
hf_chiral_get_interactions(_,Set-Ring,Senses,ChiInt,[],OHME,NO2):-
hf_chiral_pairs(Set,CAPairs),
hf_chiral_get_interactions_aux3(CAPairs,Ring,[],ChiInt),
hf_chiral_get_interactions_other(Set,Ring,Senses,ChiInt,_,OHME,NO2).

/*************************************************************************
Get the other interactions that occur in chiral atoms
*************************************************************************/
hf_chiral_get_interactions_other(Set,Ring,Senses,ChiInt,NewChiInt,
OHME,NO2):-
hf_chiral_no2_interact(Ring,Senses,ChiInt,NewChiInt,NO2),
hf_chiral_dipmult(Set,Ring,OHME).

/*********************************************************************
If in a 5 membered ring and a no2 group is bonded to an O which is
bonded to the CA's and these CAs are ortho, then this is a "true"
interaction between trans components

This is true for no2 as measured by a russian group 1985
If neither of these cases succeeds, then Chiral stays the same
and a multiplier of one is returned.
*********************************************************************/
hf_chiral_no2_interact(Ring,Senses,[],[Set],NO2):-
length(Ring,5),
hf_chiral_no2_interact_aux2(Ring,Senses,[],[Set]),
hf_chiral_no2_interact_aux(Set,NO2).
hf_chiral_no2_interact(Ring,_,[Chiral],[],NO2):-
length(Ring,5),
hf_chiral_no2_interact_aux(Chiral,NO2).
hf_chiral_no2_interact(_,_,Chiral,Chiral,1).

%%%%
hf_chiral_no2_interact_aux([_-A,_-B],NO2):-%no2
bonded(A,_,N1),
substituent_type(_,N1,no2),
bonded(B,_,N2),
substituent_type(_,N2,no2),
hf_chiral_val(no2,NO2).

%%%
hf_chiral_no2_interact_aux2([R0,R1,R2,R3,R4],[S0,S1,S2,S3,S4],In,Chiral):-
hf_chiral_no2_interact_aux2_aux([R0,R1],[S0,S1],In,C1),
hf_chiral_no2_interact_aux2_aux([R1,R2],[S1,S2],C1,C2),
hf_chiral_no2_interact_aux2_aux([R2,R3],[S2,S3],C2,C3),
hf_chiral_no2_interact_aux2_aux([R3,R4],[S3,S4],C3,C4),
hf_chiral_no2_interact_aux2_aux([R4,R0],[S4,S0],C4,Chiral).

%%%
hf_chiral_no2_interact_aux2_aux([A1,A2],[S1,S2],In,Out):-
S1 \= 0,
S2 \= 0,
S1 \= S2,
chiral_atom_info(A1,_,S1,B1,_,_),
chiral_atom_info(A2,_,S2,B2,_,_),
append([[A1-B1,A2-B2]],In,Out).
hf_chiral_no2_interact_aux2_aux(_,_,In,In).

/*******************************************************************
Gets the chiral correction for sets of chiral pairs
Finds all the chiral_pairs that have already been corrected for,
if already done - recall,
if not - make sure pairs are next to each other and same sense
then set as an interaction
*******************************************************************/
hf_chiral_get_interactions_aux3([],_,Out,Out).
hf_chiral_get_interactions_aux3([[X,Y]|T],Ring,In,Chiral):-
findall(Pair,done_chiral_pairs(Pair),DonePairs),
\+ memberchk([X,Y],DonePairs),
!,
assert(done_chiral_pairs([X,Y])),
assert(done_chiral_pairs([Y,X])),
bonded(X,_,Y),
hf_chiral_sense([X,Y],[S,S]),
hf_chiral_get_interactions_aux3(T,Ring,[[X,Y]|In],Chiral).
hf_chiral_get_interactions_aux3([_|T],Ring,In,Chiral):-
hf_chiral_get_interactions_aux3(T,Ring,In,Chiral).

/*******************************************************************
gets the 'Axial' atoms in a ring
*******************************************************************/
hf_chiral_get_interactions_aux2(_,[],[],_,Axial,Axial).
hf_chiral_get_interactions_aux2(odd,[H2|T2],[S0|T],S0,In,Axial):-
```

```
chiral_atom_info(H2,_,S0,B1,_,_),
append([H2-B1],In,Out),
hf_chiral_get_interactions_aux2(even,T2,T,S0,Out,Axial).
hf_chiral_get_interactions_aux2(odd,[_|T2],[_|T],S0,In,Axial):-
hf_chiral_get_interactions_aux2(even,T2,T,S0,In,Axial).

hf_chiral_get_interactions_aux2(even,[H2|T2],[S0|T],S0,In,Axial):-
reactophore_fact(_,_,[H2],_,S),
length(S,3),
hf_chiral_get_interactions_aux2(odd,T2,T,S0,In,Axial).
hf_chiral_get_interactions_aux2(even,[H2|T2],[S0|T],S0,In,Axial):-
reactophore_fact(_,_,[H2],_,S),
length(S,4),
S0 \= 0,
(S0 = 1 -> S1 = 2
 ; S1 = 1
),
chiral_atom_info(H2,_,S1,B1,_,_),
append([H2-B1],In,Out),
hf_chiral_get_interactions_aux2(odd,T2,T,S0,Out,Axial).

hf_chiral_get_interactions_aux2(even,[H2|T2],[H|T],S0,In,Axial):-
H \= 0,
chiral_atom_info(H2,_,H,B1,_,_),
append([H2-B1],In,Out),
hf_chiral_get_interactions_aux2(odd,T2,T,S0,Out,Axial).
hf_chiral_get_interactions_aux2(even,[_|T2],[_|T],S0,In,Axial):-
hf_chiral_get_interactions_aux2(odd,T2,T,S0,In,Axial).


/*****************************************************************
gets the 'Chiral' interactions in a ring
*****************************************************************/
hf_chiral_get_interactions_aux(0,[R0,R1],[S0,S1],In,S0,Chiral):-
hf_chiral_get_interactions_aux_aux1([R0,R1],[S0,S1],S0,In,Chiral).
hf_chiral_get_interactions_aux(3,[R0,R1,R2],[S0,S1,S2],In,S0,Chiral):-
hf_chiral_get_interactions_aux_aux1([R0,R1],[S0,S1],S0,In,C1),
hf_chiral_get_interactions_aux_aux1([R1,R2],[S1,S2],S0,C1,C2),
hf_chiral_get_interactions_aux_aux1([R2,R0],[S2,S0],S0,C2,Chiral).
hf_chiral_get_interactions_aux(4,[R0,R1,R2,R3],
[S0,S1,S2,S3],In,S0,Chiral):-
hf_chiral_get_interactions_aux_aux1([R0,R1],[S0,S1],S0,In,C1),
hf_chiral_get_interactions_aux_aux1([R1,R2],[S1,S2],S0,C1,C2),
hf_chiral_get_interactions_aux_aux1([R2,R3],[S2,S3],S0,C2,C3),
hf_chiral_get_interactions_aux_aux1([R3,R0],[S3,S0],S0,C3,C4),
hf_chiral_get_interactions_aux_aux1([R1,R3],[S1,S3],S0,C4,Chiral).
hf_chiral_get_interactions_aux(5,[R0,R1,R2,R3,R4],
[S0,S1,S2,S3,S4],In,S0,Chiral):-
hf_chiral_get_interactions_aux_aux1([R0,R1],[S0,S1],S0,In,C1),
hf_chiral_get_interactions_aux_aux1([R1,R2],[S1,S2],S0,C1,C2),
hf_chiral_get_interactions_aux_aux1([R2,R3],[S2,S3],S0,C2,C3),
hf_chiral_get_interactions_aux_aux1([R3,R4],[S3,S4],S0,C3,C4),
hf_chiral_get_interactions_aux_aux1([R4,R0],[S4,S0],S0,C4,Chiral).
hf_chiral_get_interactions_aux(6,_,[S0,_,_,_,_,_],_,S0,[]).

%%%
hf_chiral_get_interactions_aux_aux1([A1,A2],[S1,S1],S1,In,Out):-
chiral_atom_info(A1,_,S1,B1,_,main),
chiral_atom_info(A2,_,S1,B2,_,main),
append([[A1-B1,A2-B2]],In,Out).
hf_chiral_get_interactions_aux_aux1([A1,A2],[S1,S2],_,In,Out):-
reactophore_fact(_,_,[A1],_,As),
length(As,4),
S1 \= 0,
(S1 = 1 -> S2 = 2
 ; S2 = 1
),
chiral_atom_info(A1,_,S1,B1,_,main),
chiral_atom_info(A2,_,S2,B2,_,main),
append([[A1-B1,A2-B2]],In,Out).
hf_chiral_get_interactions_aux_aux1(_,_,_,In,In).


/*****************************************************************
Code for when a chain of atoms has chirality.
!!! if necessary - have to write this code
*****************************************************************/
hf_chiral_chain(_,In,In) :- !.

/*****************************************************************
RULE SECTION

-rule1:  for CAs in 1 ring
hf_chiral_rule1(RLen,Set-Ring,ChiInt,AxialInt,OHME,NO2,Chiral)

-rule2:  for CAs in a bridge structure
hf_chiral_rule2(Type,Ring,SubMult,Senses,Chiral)

-rule3:  for CAs in a sideshare structure
```

```
hf_chiral_rule3(Ring1,Ring2,SubMult,Senses,Chiral)

-rule4:  for CAs in a sideshare structure w/ multiple sides shared

*********************************************************************/
%%%%%%% RULE FOR CHAINS %%%%%%%%%
hf_chiral_rule1(0,_Set,_ChiInt,_,_,_,Chiral):-
Chiral is 0.

%%%%%%% RULE FOR SINGLE RINGS %%%%%%%%%
hf_chiral_rule1(_,Set-Ring,ChiInt,AxInt,OHME,NO2,Chiral):-
hf_chiral_rule1_aux(ChiInt,Set-Ring,AxInt,OHME,NO2,0,Chiral).

%%%%%%% ENDING RULE FOR SINGLE RINGS IF FAILS ELSEWHERE %%%%%%%%%
hf_chiral_rule1(_,_,_,_,_,_,0).


%%%%%%% RULE FOR BRIDGE SECTIONS %%%%%%%%%
hf_chiral_rule2(bridgejoin,_,_,[X,X,X],Chiral):-
hf_chiral_val(chiral,V1),
Chiral is V1.
hf_chiral_rule2(bridgejoin,_,_,[X,X,_],0).
hf_chiral_rule2(bridgejoin,_,_,[X,_,X],Chiral):-
hf_chiral_val(chiral,V1),
Chiral is V1 * 2.

hf_chiral_rule2(bridgering,_,_,[X,X],0).
hf_chiral_rule2(bridgering,_,SubMult,[_,_],Chiral):-
hf_chiral_val(chiral,V1),
Chiral is V1 * SubMult.

hf_chiral_rule2(bridgechain,_,_,[X,X],0).
hf_chiral_rule2(bridgechain,_,SubMult,[_,_],Chiral):-
hf_chiral_val(chiral,V1),
Chiral is V1 * SubMult.

%%%%%%% RULE FOR SIDESHARE SECTIONS %%%%%%%%%
hf_chiral_rule3(5,5,_,_,[X,X],0).
hf_chiral_rule3(5,5,_,_,[_,_],Chiral):-
hf_chiral_val(chiral5trans,Chiral).
hf_chiral_rule3(Short,Long,SubMult,_,[X,X],Chiral):-
Short < 5,
Long > 7,
hf_chiral_val(chiral,V1),
Chiral is V1 * SubMult.
```

```
hf_chiral_rule3(Short,Long,_,_,[_,_],0):-
Short < 5,
Long > 7.
hf_chiral_rule3(Short,Long,_,_,[X,X],0):-
Short < 5,
Long =< 7.
hf_chiral_rule3(Short,Long,SubMult,_,[_,_],Chiral):-
Short < 5,
Long =< 7,
hf_chiral_val(chiral,V1),
Chiral is V1 * SubMult.
hf_chiral_rule3(5,6,_,0,[X,X],0).
hf_chiral_rule3(5,6,SubMult,0,[_,_],Chiral):-
hf_chiral_val(chiral,V1),
Chiral is V1 * SubMult.

hf_chiral_rule3(5,6,_,_,[X,X],0).
hf_chiral_rule3(5,6,SubMult,_,[_,_],Chiral):-
hf_chiral_val(chiral,V1),
Chiral is V1 * SubMult.

hf_chiral_rule3(5,7,_,_,[_,_],0). %5-7 either direction get 0
hf_chiral_rule3(6,6,SubMult,_,[X,X],Chiral):-
hf_chiral_val(chiral6cis,V1),
Chiral is V1 * SubMult.
hf_chiral_rule3(6,6,_,_,[_,_],0).
hf_chiral_rule3(_,_,_,_,[_,_],0). %catch-all for everything else


hf_chiral_rule4(6,6,6,[[X,X],[Y,Y]],Chiral):-%2same
hf_chiral_rule4_aux(position,X,Y,Axial),  %%boat boat boat - 2 axial
Chiral is Axial * 2.
hf_chiral_rule4(6,6,6,[[X,Y],[X,Y]],0):-% all eq -all chair
X \= Y.
hf_chiral_rule4(6,6,6,[[X,Y],[Y,X]],Chiral):-%% chair boat chair
hf_chiral_rule4_aux(position,X,Y,V),
Chiral is V * 4.

hf_chiral_rule4(6,6,6,[[X,Y],[X,X]],Chiral):-%1same - 1 axial atom
hf_chiral_rule4_aux(position,X,Y,Chiral).
hf_chiral_rule4(6,6,6,[[Y,X],[X,X]],Chiral):-
hf_chiral_rule4_aux(position,X,Y,Chiral).
hf_chiral_rule4(6,6,6,[[X,X],[X,Y]],Chiral):-
hf_chiral_rule4_aux(position,X,Y,Chiral).
hf_chiral_rule4(6,6,6,[[X,X],[Y,X]],Chiral):-
```

```
hf_chiral_rule4_aux(position,X,Y,Chiral).

hf_chiral_rule4(6,6,6,[[X,X],[X,X]],0).   %allsame
%Correction determined in _aux3 %%boat boat boat


hf_chiral_rule4(_,_,_,[[X,X],[Y,Y]],0):-%2same
X \= Y.
hf_chiral_rule4(_,_,_,[[X,Y],[X,Y]],0):-
X \= Y.
hf_chiral_rule4(_,_,_,[[X,Y],[Y,X]],0):-
X \= Y.
hf_chiral_rule4(_,_,_,[[X,Y],[X,X]],0):-%1same
X \= Y.
hf_chiral_rule4(_,_,_,[[Y,X],[X,X]],0):-
X \= Y.
hf_chiral_rule4(_,_,_,[[X,X],[X,Y]],0):-
X \= Y.
hf_chiral_rule4(_,_,_,[[X,X],[Y,X]],0):-
X \= Y.
hf_chiral_rule4(4,_,_,[[X,X],[X,X]],Chiral):-  %allsame
hf_chiral_val(chiral,V1),
Chiral is V1 * 4.
hf_chiral_rule4(_,_,_,[[X,X],[X,X]],Chiral):-  %allsame
hf_chiral_val(chiral,V1),
Chiral is V1 * 2.

%%%
hf_chiral_rule4_aux(AA,X,Y,Chiral):-
X \= Y,
hf_chiral_val(AA,V2),
Chiral is V2.

/*******************************************************************
hf_chiral_rule1_aux(Set,Set-Ring,AxInt,OHME,NO2,SubMult,Chiral)
Set is the set of chiral interactions
AxInt is the list of Axial CAs
Special is the multiplier caused by special interactions
no2 in a 5
oh-methyl off a ring
*******************************************************************/
hf_chiral_rule1_aux([],_,AxInt,OHME,NO2,SubMult,Chiral):-
hf_chiral_rule1_aux2(AxInt,0,AxMult),
hf_chiral_val(chiral,V1),
hf_chiral_val(position,V2),
Chiral is (V1 * SubMult * NO2) + (V2 * AxMult) + OHME.
hf_chiral_rule1_aux([[CA1,CA2]|T],Set-Ring,AxInt,OHME,NO2,Part,Chiral):-
hf_chiral_size(CA1,SubMult1),
hf_chiral_size(CA2,SubMult2),
NewPart is Part + (SubMult1 + SubMult2) / 2,
hf_chiral_rule1_aux(T,Set-Ring,AxInt,OHME,NO2,NewPart,Chiral).

hf_chiral_rule1_aux2([],Part,Part).
hf_chiral_rule1_aux2([H|T],InP,Part):-
hf_chiral_size(H,AxMult),
NewPart is InP + AxMult,
hf_chiral_rule1_aux2(T,NewPart,Part).


/*****************************************************************
hf_chiral_size
gets the chiral multiplier;
dependant on the size of the chiral Substituent
*****************************************************************/
hf_chiral_size(CA-BA,SubMult):-
chiral_atom_info(CA,_,_,BA,Size,_),
(0.05 < Size, Size < 0.11 -> Size1 = 0.05
   ; Size1 = Size
),
SubMult is Size1 / 0.0506424308.

/*****************************************************************
Returns the sets of chiral atoms and asserts
them in the database
*****************************************************************/
hf_chiral_pairs(CAs,Pairs):-
findall([X,Y],(member(X,CAs),
       member(Y,CAs),
       X \= Y,
       assert(chiral_pairs([X,Y]))
      ),Pairs).

/*****************************************************************
Gets the shortest distance between atoms in a ring
*****************************************************************/
hf_atom_distance(X,Y,Dist):-
reactophore_fact(ReacX,_,AtomsX,_,_),
memberchk(X,AtomsX),
reactophore_fact(ReacY,_,AtomsY,_,_),
memberchk(Y,AtomsY),
all_paths(ReacX,APs),
get_path_aux(ReacY,Y,X,APs,[_,DistL,_]),
minimum(DistL,Dist).

/*******************************************************************
In a single ring if a methyl and oh have same sense
(except for meta, then opposite), then need an additonal correction

hf_chiral_dipmult(Set,Ring,OHME)
OHME is the correction dependent on the Subs (off of each CA)

_get the OHs
examine each oh for being near a methyl, get total
number of interactions and mult by Val
*******************************************************************/
hf_chiral_dipmult(Set,Ring,OHME):-
hf_chiral_atom_subs(Set,Ring,Subs),
% hf_chiral_breaksubs(Subs,NewSubs1),
hf_chiral_get_ohs(Subs,NewSubs),
hf_chiral_dipmult_aux_top(NewSubs,Subs,Ring,0,NumInts),
hf_chiral_val(oh-methyl,Val),
OHME is Val * NumInts.
hf_chiral_dipmult(_,_,0).

%%%%
hf_chiral_dipmult_aux_top([],_,_,Out,Out).
hf_chiral_dipmult_aux_top([Sub|Rest],NewSubs1,Ring,Part,TotalInts):-
hf_chiral_dipmult_aux(Sub,NewSubs1,Ring,NumInts),
NewPart is Part + NumInts,
hf_chiral_dipmult_aux_top(Rest,NewSubs1,Ring,NewPart,TotalInts).

%%%%
hf_chiral_dipmult_aux(Sub,Subs,Ring,NumInts):-
length(Ring,LenRing),
LenRing = 6,
Sub = [Y-CA],
delete(Subs,Sub,Rest),
chiral_atom_info(CA,_,SOH,Y,_,_),
hf_chiral_dipmult_12s(Rest,CA,SOH,NumOs),
hf_chiral_dipmult_13s(Rest,CA,SOH,NumMs),
hf_chiral_dipmult_14s(Rest,CA,SOH,NumPs),
NumInts is NumOs + NumMs + NumPs.
hf_chiral_dipmult_aux(Sub,Subs,Ring,NumInts):-
length(Ring,LenRing),
LenRing = 4,
Sub = [Y-CA],
delete(Subs,Sub,Rest),
chiral_atom_info(CA,_,SOH,Y,_,_),
hf_chiral_dipmult_12s(Rest,CA,SOH,NumOs),
hf_chiral_dipmult_13s(Rest,CA,SOH,NumMs),
NumInts is NumOs + NumMs.
hf_chiral_dipmult_aux(Sub,Subs,_,NumInts):-
Sub = [Y-CA],
delete(Subs,Sub,Rest),
chiral_atom_info(CA,_,SOH,Y,_,_),
hf_chiral_dipmult_12s(Rest,CA,SOH,NumInts).
hf_chiral_dipmult_aux(_,_,_,0).

%%%
hf_chiral_dipmult_12s(Rest,CA,SOH,NumOs):-
findall(B,(member([B-CA1],Rest),
   bonded(CA,_,CA1),
   chiral_atom_info(CA1,_,SOH,B,_,_),
   substituent_type(_,B,methyl)),ListO1s),
delete_duplicates(ListO1s,ListOs),
length(ListOs,NumOs).
%%%
hf_chiral_dipmult_13s(Rest,CA,SOH,NumMs):-
findall(B,(member([B-CA1],Rest),
   bonded(CA,_,A),
   bonded(A,_,CA1),
   CA1 \= CA,
   chiral_atom_info(CA1,_,S,B,_,_),
   S \= SOH,
   substituent_type(_,B,methyl)),ListM1s),
delete_duplicates(ListM1s,ListMs),
length(ListMs,NumMs).
%%%
hf_chiral_dipmult_14s(Rest,CA,SOH,NumPs):-
findall(B,(member([B-CA1],Rest),
   bonded(CA,_,A1),
   bonded(A1,_,A2),
   A2 \= CA,
   bonded(A2,_,CA1),
   CA1 \= A1,
   chiral_atom_info(CA1,_,SOH,B,_,_),
   substituent_type(_,B,methyl)),ListP1s),
delete_duplicates(ListP1s,ListPs),
length(ListPs,NumPs).
```

```
/****
%%%%
hf_chiral_breaksubs([],[]).
hf_chiral_breaksubs([[A-CA]|T],[[A-CA]|NewSubs]):-
hf_chiral_breaksubs(T,NewSubs).
hf_chiral_breaksubs([[[A,B]-CA]|T],[[[A]-CA],[[B]-CA]|NewSubs]):-
hf_chiral_breaksubs(T,NewSubs).

***/

%%%%
hf_chiral_get_ohs([],[]).
hf_chiral_get_ohs([Set|T],[Set|NewSubs]):-
Set = [X-_],
substituent_type(_,X,oh),
hf_chiral_get_ohs(T,NewSubs).
hf_chiral_get_ohs([_|T],NewSubs):-
hf_chiral_get_ohs(T,NewSubs).

/*********************************************************************
take a (list of lists of lists) and convert to a (list of lists)
*********************************************************************/
lol3_to_lol2([],Out,Out).
lol3_to_lol2([H|T],In,ListRings):-
append(H,In,NewIn),
lol3_to_lol2(T,NewIn,ListRings).

/*********************************************************************
removes [H|T] from CASets
*********************************************************************/
hf_chiral_remove_sets([],Out,Out).
hf_chiral_remove_sets([H|T],CASets,RestSets):-
delete(CASets,H,Rest),
hf_chiral_remove_sets(T,Rest,RestSets).


/************** VVVVVVVVVVVVVVVVVVVVVVVVVVV ***************************/

/*********************************************************************
Takes these Sets and Rings and builds the fact:
chiral_atom_info(CA,Ring,Sense,Sub,SubSize,Type)
*********************************************************************/
hf_chiral_set_data_sideshare(SideSets,SetRings):-
hf_chiral_set_senses(SideSets,SetRings,SideSets,Senses),
hf_chiral_set_data_bridge_aux(SideSets,Senses).  %(well this might work)
```

```
/****************************************************************
Takes these Sets and Rings and builds the fact:
chiral_atom_info(CA,Ring,Sense,Sub,SubSize,Type)
****************************************************************/
hf_chiral_set_data_bridge(NewBridgeSets,SetRings):-
hf_chiral_set_senses(NewBridgeSets,SetRings,NewBridgeSets,Senses),
hf_chiral_set_data_bridge_aux(NewBridgeSets,Senses).

%%%
hf_chiral_set_data_bridge_aux([],_).
hf_chiral_set_data_bridge_aux([Set-Ring|Rest],[RingSenses|RSs]):-
hf_chiral_subsize(Set,Ring,SizeData),
hf_chiral_data_bridge_aux_aux(RingSenses,Senses),
hf_chiral_set_subsense(Ring,Senses,SizeData,SenseData),
hf_chiral_assert_fact(SenseData,Ring),
hf_chiral_set_data_bridge_aux(Rest,RSs).
hf_chiral_set_data_bridge_aux([_|Rest],Senses):-  %should never get here
abort,
hf_chiral_set_data_bridge_aux(Rest,Senses).

%%%
hf_chiral_data_bridge_aux_aux([],[]).
hf_chiral_data_bridge_aux_aux([[H,_,S]|T],[[S,H]|Senses]):-
hf_chiral_data_bridge_aux_aux(T,Senses).

/****************************************************************
  Set the CA senses relative to whole molecule
****************************************************************/
hf_chiral_set_senses([],_,BSets,Senses):-
!,
hf_chiral_set_senses_aux3(BSets,Senses).
hf_chiral_set_senses([Set-Ring|T],SetRings,BSets,Senses):-
delete(SetRings,Ring,Rest),
hf_chiral_set_senses_aux1(Set,Rest,OtherRing),
hf_chiral_set_sense(Ring,Set,Ring,OtherRing,S1),
hf_chiral_set_senses_aux(Ring,Ring,S1),
hf_chiral_set_senses(T,SetRings,BSets,Senses).

/****************************************************************
Finds which ring H is in, b/c that'll be OtherRing
****************************************************************/
hf_chiral_set_senses_aux1([],_,[]).
hf_chiral_set_senses_aux1([H|_],RestRings,OtherRing):-
member(OtherRing,RestRings),
memberchk(H,OtherRing).
hf_chiral_set_senses_aux1([_|T],RestRings,OtherRing):-
hf_chiral_set_senses_aux1(T,RestRings,OtherRing).

/*****************************************************************
asserts the chiral_sense fact in the DB
*****************************************************************/
hf_chiral_set_senses_aux([],_,[]).
hf_chiral_set_senses_aux([H|T],Ring,[[S1,_]|T1]):-
findall(Sense,chiral_sense(H,Ring,Sense),[]),%not in - assert it
assert(chiral_sense(H,Ring,S1)),
hf_chiral_set_senses_aux(T,Ring,T1).
hf_chiral_set_senses_aux([H|T],Ring,[[S1,_]|T1]):-
findall(Sense,chiral_sense(H,Ring,Sense),[S1]),
hf_chiral_set_senses_aux(T,Ring,T1).
hf_chiral_set_senses_aux([H|T],Ring,[[S1,_]|T1]):-
findall(Sense,chiral_sense(H,Ring,Sense),[Sense]),
assert(chiral_sense(H,Ring,S1)),
hf_chiral_set_senses_aux(T,Ring,T1).

/*****************************************************************
real work done here for making the molecule the same
*****************************************************************/
hf_chiral_set_senses_aux3([],[]).
hf_chiral_set_senses_aux3([_-Ring|T],[S|Senses]):-
hf_chiral_set_senses_aux3_aux(Ring,Ring),
hf_chiral_set_senses_aux3_aux2(Ring,Ring,S),
hf_chiral_set_senses_aux3(T,Senses).

%%%
hf_chiral_set_senses_aux3_aux2([],_,[]).
hf_chiral_set_senses_aux3_aux2([H|T],Ring,[[H,Ring,S0]|S]):-
chiral_sense(H,Ring,S0),
hf_chiral_set_senses_aux3_aux2(T,Ring,S).
hf_chiral_set_senses_aux3_aux2([_|T],Ring,S):-%should never get here
hf_chiral_set_senses_aux3_aux2(T,Ring,S).

/*****************************************************************
Finds out if the Ring you are examining has a CA set to a different
sense for a different ring, if it does then fix that other ring
*****************************************************************/
hf_chiral_set_senses_aux3_aux([],_).
hf_chiral_set_senses_aux3_aux([H|T],Ring):-
chiral_sense(H,Ring,Sense),
findall(OR,(chiral_sense(H,OR,S1),
    Sense \= S1),[OR]),
findall([Atom,OR,S2],chiral_sense(Atom,OR,S2),List),
hf_chiral_set_senses_aux3_aux_aux(List),
hf_chiral_set_senses_aux3_aux(T,Ring).
hf_chiral_set_senses_aux3_aux([_|T],Ring):-
hf_chiral_set_senses_aux3_aux(T,Ring).

/*****************************************************************
Sets the other rings sense to the opposite of what it was
*****************************************************************/
hf_chiral_set_senses_aux3_aux_aux([]).
hf_chiral_set_senses_aux3_aux_aux([[_,_,0]|T]):-
!,
hf_chiral_set_senses_aux3_aux_aux(T).
hf_chiral_set_senses_aux3_aux_aux([[Atom,OR,S2]|T]):-
retract(chiral_sense(Atom,OR,S2)),
(S2 =:= 1 -> NewSense = 2
   ; NewSense = 1
),
assert(chiral_sense(Atom,OR,NewSense)),
hf_chiral_set_senses_aux3_aux_aux(T).

/*****************************************************************
Takes a Set and a Ring and builds the fact:
chiral_atom_info(CA,Ring,Sense,Sub,SubSize,Type)

_subsize: returns [[CA,SubAtom,SizeofSub],...] for each sub
off of CA, not in ring
_set_sense: find senses of RingAtoms - [[0,A1],[1,A2],[2,A3],...]
_ring_rot: rotates Ring so largest sub is first
_set_subsense: gets the relative senses of the ring
_assert_fact: asserts the chiral_atom_info fact
*****************************************************************/
hf_chiral_set_data_single(Set,Ring,NewRing):-
hf_chiral_subsize(Set,Ring,SizeData),
hf_chiral_set_sense(Ring,Set,Ring,Ring,Senses),
hf_chiral_ring_rot(Ring,SizeData,Senses,NewRing),
hf_chiral_set_subsense(NewRing,Senses,SizeData,SenseData),
hf_chiral_assert_fact(SenseData,NewRing).

%%%%
hf_chiral_assert_fact([],_).
hf_chiral_assert_fact([[CA,Sense,Sub,SubSize,Type]|T],Ring):-
assert(chiral_atom_info(CA,Ring,Sense,Sub,SubSize,Type)),
hf_chiral_assert_fact(T,Ring).
```

```
/*******************************************************************
determines the senses of the subatoms
*******************************************************************/
hf_chiral_set_subsense([],_,_,[]).
hf_chiral_set_subsense([RA|Rest],Senses,SizeData,[[RA,0,_,_,_]|SenseData]):-
findall([RA,SubAtom,Size],member([RA,SubAtom,Size],SizeData),[]),
hf_chiral_set_subsense(Rest,Senses,SizeData,SenseData).
hf_chiral_set_subsense([RA|Rest],Senses,SizeData,
[[RA,S,SA,Size,main]|SenseData]):-
findall([RA,SA,Size],member([RA,SA,Size],SizeData),[[RA,SA,Size]]),
memberchk([S,RA],Senses),
hf_chiral_set_subsense(Rest,Senses,SizeData,SenseData).
hf_chiral_set_subsense([RA|Rest],Senses,SizeData,
       [[RA,S1,SA1,Size1,Type1],[RA,S2,SA2,Size2,Type2]|SenseData]):-
findall([RA,SA,Size],member([RA,SA,Size],SizeData),
 [[RA,SA1,Size1],[RA,SA2,Size2]]),
memberchk([S,RA],Senses),
(S = 1 -> OS = 2
; OS = 1
),
(Size1 > Size2 -> S1 = S,
  S2 = OS,
  Type1 = main,
  Type2 = sec
; S2 = S,
  S1 = OS,
  Type1 = sec,
  Type2 = main
),
hf_chiral_set_subsense(Rest,Senses,SizeData,SenseData).

/*******************************************************************
gets the senses of all the Atoms in a ring
if it's not a CA then put sense to 0
the output is in the same order as the incoming list
returns [[SenseA,A],....]
*******************************************************************/
hf_chiral_set_sense([],_,_,_,[]).
hf_chiral_set_sense([A|Rest],Set,Ring,OtherRing,[[SenseA,A]|Senses]):-
memberchk(A,Set),
hf_chiral_sense(A,Ring,OtherRing,SenseA),
hf_chiral_set_sense(Rest,Set,Ring,OtherRing,Senses).
hf_chiral_set_sense([A|Rest],Set,Ring,OtherRing,[[0,A]|Senses]):-
hf_chiral_set_sense(Rest,Set,Ring,OtherRing,Senses).
```

```
/*****************************************************************
/2      Takes a list of atoms and brings back the senses

/4 Figures out the "sense" of the chiral atoms: 1 or 2

_ring:  get_chirality(CA,InAtom,Outs)
   CA is the ChiralCenter,
   In is the InAtom,
   Outs are the OutAtoms
_chain: not done because not used anywhere at this time (6/04)
*****************************************************************/
hf_chiral_sense([],[]).
hf_chiral_sense([H|T],[S|Senses]) :-
chiral_atom_info(H,_,S,_,_,main),
hf_chiral_sense(T,Senses).

hf_chiral_sense(CA1,Ring,OtherRing,SenseCA1) :-
hf_chiral_sense_ring(CA1,Ring,OtherRing,SenseCA1).
hf_chiral_sense(CA1,Ring,OtherRing,SenseCA1) :-
hf_chiral_sense_chain(CA1,Ring,OtherRing,SenseCA1).

%%%
hf_chiral_sense_chain(_CA1,_Ring,_OtherRing,SenseCA1):-%have to write
SenseCA1 is 1.

%%%
hf_chiral_sense_ring(CA,Ring,OtherRing,Sense):-
hf_get_inatom(CA,Ring,InAtom),
reactophore_fact(_,_,[CA],_,Subs),
delete(Subs,[InAtom],RestSubs),
findall(X,(member([X],RestSubs),
   member(X,Ring)),[RingSub]),
delete(RestSubs,[RingSub],LeftOver),
hf_get_regsub(LeftOver,CA,OtherRing,RegSub),
Outs = [RegSub,RingSub],
get_chirality(CA,InAtom,Outs,Sense).

%%
hf_get_inatom(CA,Ring,InAtom):-
nth(Ring,Pos,CA),
PrevPos is Pos - 1,
(PrevPos > 0 ->nth(Ring,PrevPos,InAtom)
      ; reverse(Ring,RevRing),
nth(RevRing,1,InAtom)
).

%%%
hf_get_regsub([[RegSub1]],_,_,RegSub1) :- !.
hf_get_regsub([[RegSub1],[_RegSub2]],_,OtherRing,RegSub1) :-
memberchk(RegSub1,OtherRing),
!.
hf_get_regsub([[_RegSub1],[RegSub2]],_,OtherRing,RegSub2) :-
memberchk(RegSub2,OtherRing),
!.
hf_get_regsub([[RegSub1],[RegSub2]],CA,_,Sub) :-
reactophore_fact(CASub,_,[CA],_,_),
reactophore_fact(Sub1,_,A1,_,_),
memberchk(RegSub1,A1),
reactophore_fact(Sub2,_,A2,_,_),
memberchk(RegSub2,A2),
hf_get_sub_size(CASub,[Sub1,Sub2],[Size1,Size2]),
(Size1 > Size2 -> Sub = RegSub1
; Sub = RegSub2
),
!.

/*******************************************************************
rotates the ring so that the CA w/ the largest substituent is first
*******************************************************************/
hf_chiral_ring_rot(Ring,SizeData,Senses,NewRing):-
hf_chiral_set_rot_aux(Ring,SizeData,Senses,First),
hf_chiral_ring_rot_aux2(Ring,First,NewRing).

%%%%
hf_chiral_ring_rot_aux2([H|T],H,[H|T]).
hf_chiral_ring_rot_aux2([H|T],First,NewRing):-
append(T,[H],RRing),
hf_chiral_ring_rot_aux2(RRing,First,NewRing).

/*******************************************************************
gets the first element in the ring
*******************************************************************/
hf_chiral_set_rot_aux(Ring,SizeData,Senses,First):-
findall(Size,(member([A,_,Size],SizeData),
      memberchk(A,Ring)),Sizes1),
delete_duplicates(Sizes1,Sizes),
maximum(Sizes,Size),
findall(CA,member([CA,_,Size],SizeData),Potentials),
hf_chiral_set_rot_aux_aux1(Potentials,SizeData,Bs),
hf_chiral_set_rot_aux_aux2(Bs,Potentials,Size,SizeData,Ring,
 Senses,[First|_]).

/*****************************************************************
gets the number of branches off of each potential first CA
*****************************************************************/
hf_chiral_set_rot_aux_aux1([],_,[]).
hf_chiral_set_rot_aux_aux1([H|T],SizeData,[Bs|In]):-
findall(B,member([H,B,_],SizeData),ListBs),
length(ListBs,Bs),
hf_chiral_set_rot_aux_aux1(T,SizeData,In).

/*****************************************************************
Depending on the number of branches gets correction
*****************************************************************/
hf_chiral_set_rot_aux_aux2(Bs,Pots,Size,SizeData,Ring,Senses,NewPots):-
%for Pots w/ 1 and 2Bs
memberchk(2,Bs),
memberchk(1,Bs),
findall(CA,(member(CA,Pots),
    findall(X,member([CA,_,X],SizeData),[_])),CA1s),
difference(Pots,CA1s,CA2s),
hf_chiral_set_rot_aux_aux2([2],CA2s,Size,Ring,Senses,NewPots).
hf_chiral_set_rot_aux_aux2(Bs,Pots,Size,SizeData,_,_,NewPots):-
%for Pots w/ 2Bs
memberchk(2,Bs),
findall(X,(member(CA,Pots),
   member([CA,_,X],SizeData),
   X \= Size),Sizes),
maximum(Sizes,BSize),
member([First,_,BSize],SizeData),
member([First,_,Size],SizeData),
difference(Pots,[First],Rest),
NewPots = [First|Rest].
hf_chiral_set_rot_aux_aux2(_,Pots,_,_,Ring,S1,NewPots):-%for Pots w/ 1B
findall(Sense,(member(CA,Pots),
       memberchk([Sense,CA],S1)),Senses),
findall(1,member(1,Senses),ListOnes),
length(ListOnes,NumOnes),
findall(2,member(2,Senses),ListTwos),
length(ListTwos,NumTwos),
hf_chiral_set_rot_aux_aux2_aux(NumOnes,NumTwos,Pots,Ring,S1,NewPots).

/*****************************************************************
Gets Potentials based on number of senses in a certain direction
```

```
-more 1s than 2s, get the 1s
-more 2s than 1s, get the 2s
-same number of 1s and 2s, filter
******************************************************************/
hf_chiral_set_rot_aux_aux2_aux(NumOnes,NumTwos,Pots,Ring,Senses,NewPots):-
NumOnes > NumTwos,
findall(CA,(member(CA,Pots),
    memberchk([1,CA],Senses)),PPots),
hf_chiral_set_rot_aux_aux2_ring1(PPots,Ring,PotRings),
hf_chiral_set_rot_aux_aux2_ring2(PotRings,Senses,PotRings2),
hf_chiral_set_rot_aux_aux2_ring3(PotRings2,NewPots).

hf_chiral_set_rot_aux_aux2_aux(NumOnes,NumTwos,Pots,_,Senses,NewPots):-
NumOnes < NumTwos,
findall(CA,(member(CA,Pots),
    memberchk([2,CA],Senses)),NewPots).
hf_chiral_set_rot_aux_aux2_aux(Num,Num,Pots,Ring,Senses,NewPots):-
findall(CA,(member(CA,Pots),
    member([1,CA],Senses)),PPots),
hf_chiral_set_rot_aux_aux2_ring1(PPots,Ring,PotRings),
hf_chiral_set_rot_aux_aux2_ring2(PotRings,Senses,PotRings2),
hf_chiral_set_rot_aux_aux2_ring3(PotRings2,NewPots).
hf_chiral_set_rot_aux_aux2_aux(_,_,_,_,_,[]). %Should never be here

/******************************************************************
take all of the potentials and get the potential rings from them
******************************************************************/
hf_chiral_set_rot_aux_aux2_ring1([],_,[]).
hf_chiral_set_rot_aux_aux2_ring1([H|T],Ring,[NewRing|PotRings]):-
hf_chiral_ring_rot_aux2(Ring,H,NewRing),
hf_chiral_set_rot_aux_aux2_ring1(T,Ring,PotRings).

/******************************************************************
take all of the potential rings
- if 6 and senses are one of 5 cases, throw away ring  (ghetto way of doing)
- else keep pot rings
******************************************************************/
hf_chiral_set_rot_aux_aux2_ring2([],_,[]).
hf_chiral_set_rot_aux_aux2_ring2([Ring|Rest],Senses,NewPots):-
length(Ring,6),
findall(S,(member(A,Ring),
   member([S,A],Senses)),RSenses),
hf_chiral_set_rot_aux_aux2_ring2_aux(RSenses),
hf_chiral_set_rot_aux_aux2_ring2(Rest,Senses,NewPots).
hf_chiral_set_rot_aux_aux2_ring2([Ring|Rest],Senses,[Ring|NewPots]):-
```

```
hf_chiral_set_rot_aux_aux2_ring2(Rest,Senses,NewPots).

hf_chiral_set_rot_aux_aux2_ring2_aux([1,2,2,1,2,1]).
hf_chiral_set_rot_aux_aux2_ring2_aux([1,2,2,1,1,2]).
hf_chiral_set_rot_aux_aux2_ring2_aux([1,2,1,1,2,2]).
hf_chiral_set_rot_aux_aux2_ring2_aux([1,1,2,2,2,1]).

hf_chiral_set_rot_aux_aux2_ring2_aux([1,1,2,1,2,0]).

/*****************************************************************
all else being equal, take the first atom of the first pot ring and
that's the new pot
*****************************************************************/
hf_chiral_set_rot_aux_aux2_ring3([[H|_]|_],[H]).

/****************************************************************
hf_chiral_subsize
returns : [[CA,SubAtom,Size],...]
****************************************************************/
hf_chiral_subsize(Set,Ring,SizeData):-
hf_chiral_atom_subs(Set,Ring,Subs),
hf_chiral_subsize_aux(Subs,Ring,SizeData).

hf_chiral_subsize_aux([],_,[]).
hf_chiral_subsize_aux([[SA-CA]|Rest],Ring,[[CA,SA,Size]|SizeData]):-
reactophore_fact(Sub,_,[CA],_,_),
reactophore_fact(ExSub,_,ExAtoms,_,_),
memberchk(SA,ExAtoms),
hf_get_sub_size(Sub,[ExSub],[Size]),
hf_chiral_subsize_aux(Rest,Ring,SizeData).

/****************************************************************
Returns the External Subs off of each CA
[[Sub1-CA,Sub2-CA],...] or  (gets large one first - need this part?)
[[Sub1-CA],...]
****************************************************************/
hf_chiral_atom_subs([],_,[]).
hf_chiral_atom_subs([CA|T],Ring,[[A1-CA]|ExSubs]):-
reactophore_fact(_,_,[CA],_,Atoms),
flatten(Atoms,FAtoms),
difference(Ring,[CA],RestRAtoms),
difference(FAtoms,RestRAtoms,[A1]),
hf_chiral_atom_subs(T,Ring,ExSubs).
hf_chiral_atom_subs([CA|T],Ring,[[A1-CA],[A2-CA]|ExSubs]):-
reactophore_fact(CASub,_,[CA],_,Atoms),
flatten(Atoms,FAtoms),
difference(Ring,[CA],RestRAtoms),
difference(FAtoms,RestRAtoms,[A1,A2]),
  reactophore_fact(Sub1,_,[A1],_,_),
reactophore_fact(Sub2,_,[A2],_,_),
hf_get_sub_size(CASub,[Sub1,Sub2],[Size1,Size2]),
(Size1 > Size2 -> Ex1 = A1, Ex2 = A2
; Ex1 = A2, Ex2 = A1
),
hf_chiral_atom_subs(T,Ring,ExSubs).
hf_chiral_atom_subs([_|T],Ring,ExSubs):-%this should never fire
hf_chiral_atom_subs(T,Ring,ExSubs).
```

C.5 HF_DATA.PRO

```
/************** DYNAMIC AREA **************/
:- dynamic
arom_ring_side/2,
hf_ba_breg/2,
hf_ba_inn/2,
hf_ba_mid/2,
hf_ba_out/2,
hf_ba_reg/2,
hf_branch_acetylenic/2,
hf_branch_ethylenic/2,
hf_branch_methyl/2,
hf_bridge/2,
hf_bridgedb/2,
hf_chiral_val/2,
hf_cis1_resonance/2,
hf_erc/2,
hf_near_methylparam/2,
hf_ortho/2,
hf_perri_param/2,
hf_rc/2,
hf_relsz/2,
hf_ring/2,
hf_ring_db/2,
hf_ring_exo/2,
hf_ring_side/2,
hf_ring_side3/2,
hf_sa/2,
hf_steric_acetylenic/2,
hf_steric_ethylenic/2,
hf_steric_ethylenic_2/2,
hf_steric_methyl/2,
hf_resonance/2,
resonance2/2,
resonance3/2,
ring_vertices_1ring/2,
ring_vertices_2ring/2,
ring_vertices_3ring/2,
ring_vertices_4ring/2,
hf_dipole/2,
hf_branch_oethylenic/2,
res_contrib/2,
hf_steric_oethylenic/2,
hf_ring_exo_oethylenic/2,
hf_steric_ring/2,
hf_ring_deform/2,
hf_contrib/2,
hf_halo/2,
hf_connect_co2h/2,
hf_halo_xi/2,
hf_halo_ethy/2,
hf_halo_ethy1/2,
hf_halo_ethy2/2,
hf_halo_ethy3/2.
```

```
/************** DYNAMIC AREA **************/
/***************************************************************************
This file is for storing the parameters that the HF code uses
```

```
The first part came from hf.pro

The second part came from hf_ring2.pro

The third part came from other_arom.pro

The fourth part is for oxygen cmpds

Created 12/2/2003 by Tad Whiteside at the behest of Raj because he
was bitching that predicates kept getting redefined elsewhere
**************************************************************************/

/**** Data Area ************************************************************/
%@@##

hf_contrib(methyl, -74.87).  %methane
hf_contrib(ethylenic,52.47).  %ethane
hf_contrib(acetylenic,226.73). %acetylene

hf_contrib(oh,-241.826).  %water
hf_contrib(co2h,-352.3).
hf_contrib(oeth,-168.37).               %formaldehyde
hf_contrib([methyl-oeth],-21.9586).
hf_contrib([methyl-oh],-21.9586).

hf_contrib(nr2,-45.94). %ammonia
hf_contrib(cn,135.14).

hf_contrib([methyl-nr2],-21.9586).

hf_contrib(f-methyl,-202.183).  % [-204.555] [2004,9,19]
hf_contrib(f-aromatic,-207.42).  % [-207.395] [2004,9,19]
hf_contrib(f-ethylenic,-196.2).  % [-197.427] [2004,9,19]
hf_contrib(f-acetylenic,-200).

hf_contrib(cl-methyl,-48.072).  % [-48.0984] [2004,9,19]
hf_contrib(cl-aromatic,-38.2942).  % [-38.2958] [2004,9,19]
hf_contrib(cl-ethylenic,-39.2465).  % [-39.1738] [2004,9,19]
hf_contrib(cl-acetylenic,-40).

hf_contrib(br-methyl,5.59636).  % [5.60728] [2004,9,19]
hf_contrib(br-aromatic,18.8797).  % [18.896] [2004,9,19]
hf_contrib(br-ethylenic,16.737).  % [16.7309] [2004,9,19]
hf_contrib(br-acetylenic,10).
hf_contrib(i-methyl,58.8433).  % [58.8394] [2004,9,19]
hf_contrib(i-aromatic,76.0191).  % [76.0828] [2004,9,19]
hf_contrib(i-ethylenic,66.4468).  % [66.4041] [2004,9,19]
hf_contrib(i-acetylenic,60).

hf_contrib(pr2,22.89).  %phosphine

hf_contrib(sh,-20.50).  %hydrogen sulfide



hf_branch_ethylenic(pp,9.99428).  % [9.92152] [2004,4,29]

hf_branch_acetylenic(a,-1.55515).  % [-1.19295] [2004,4,29]
hf_branch_acetylenic(b,6.0364).  % [6.12079] [2004,4,29]
hf_branch_acetylenic(endo,55.6573).  % [55.9945] 2004/3/9

hf_ba_reg(aromatic,10.756).  % [10.7595] 2004/3/9
hf_ba_breg(aromatic,15.5441).  % [15.5647] 2004/3/9
hf_ba_inn(aromatic,32.8096).  % [32.8169] 2004/3/9
hf_ba_out(aromatic,17.4823).  % [17.4933] 2004/3/9
hf_ba_mid(aromatic,43.3601).  % [43.3502] 2004/3/9

hf_steric_methyl(0,1).
hf_steric_methyl(1,7.16569).  % [7.24905] 2004/3/9
hf_steric_methyl(2,0.0903988).  % [0.218187] 2004/3/9
hf_steric_methyl(3,16.9948).  % [16.9831] 2004/3/9
hf_steric_methyl(4,107.251).  % [107.244] 2004/3/9

hf_steric_ethylenic(pp,10.6416).  % [7.02299] [2004,4,29]
hf_steric_ethylenic_2(pp,3.6846).  % [3.43581] [2004,4,29]

hf_steric_acetylenic(ace,-7.38874).  % [-27.5209] [2004,4,29]

hf_sa(aromatic,-11.6831).  % [-11.6591] 2004/3/9

hf_near_methylparam(4,-5.66162) :- !.  % [-5.66681] 2004/3/9

hf_ortho(inner,2.54594).  % [2.5318] 2004/3/9
hf_ortho(outer,-1.47037).  % [-1.47867] 2004/3/9
hf_ortho(inner_outer_size,66.7918).  % [66.8904] 2004/3/9
hf_ortho(outer-methyl,1).
hf_ortho(inner-methyl,1).
hf_ortho(outer-cl,0.703).  % [0.700983] [2004,9,19]
hf_ortho(inner-cl,2.48292).  % [2.48316] [2004,9,19]
hf_ortho(outer-f,-8.84239).  % [-8.78998] [2004,9,19]
hf_ortho(outer-aromaticf,7.99081).  % [8.03933] [2004,9,19]


hf_ortho(inner-f,8.85696).  % [8.8729] [2004,9,19]
hf_ortho(inner-aromaticf,26.0387).  % [25.8974] [2004,9,19]
hf_ortho(inner-fmethyl,-3.0034).  % [-3.0265] [2004,9,19]
hf_ortho(inner-methylf3,35.9336).  % [35.8597] [2004,9,19]
hf_ortho(inner-clmethyl,-1.26146).  % [-1.26266] [2004,9,19]
hf_ortho(inner-clethylenic,3.66468).  % [3.66763] [2004,8,11]
hf_ortho(inner-faromaticf,46.0414).  % [46.0311] [2004,8,11]
hf_ortho(multf,22.2226).  % [22.1987] [2004,8,11]
hf_ortho(inner-clf,-0.037185).  % [-0.0897409] [2004,9,19]
hf_ortho(inner-fi,15.1321).  % [15.1333] [2004,9,19]


hf_perri_param(aromatic,176.806).  % [176.624] 2004/3/9

hf_ring(3,113.535).  % [113.404] 2004/3/9
hf_ring(4,103.995).  % [103.836] 2004/3/9
hf_ring(5,23.4315).  % [23.3011] 2004/3/9
hf_ring(mp_b,1.00071).  % [1.00119] 2004/3/9
hf_ring(mp_d,52.3).
hf_ring(ed_a,2876.18).  % [2752.46] 2004/3/9
hf_ring(ed_b,0.366596).  % [0.362327] 2004/3/9
hf_ring(point3,16.7037).  % [16.8433] 2004/3/9
hf_ring(point,7.37964).  % [7.51005] 2004/3/9

hf_ring_db(3,131.844).  % [131.845] [2004,5,9]
hf_ring_db(4,42.035).  % [42.0342] [2004,5,9]
hf_ring_db(5,24.4884).  % [24.3075] [2004,5,9]
hf_ring_db(6,22.5003).  % [22.5001] [2004,5,9]
hf_ring_db(7,21.1901).  % [21.0001] [2004,5,9]
hf_ring_db(8-cis,21.827).  % [21.827] [2004,5,9]
hf_ring_db(8-trans,68.0094).  % [68.0093] [2004,5,9]
hf_ring_db(other-cis,14.9887).  % [14.9891] [2004,5,9]
hf_ring_db(other-trans,2.82504).  % [2.82333] [2004,5,9]

hf_ring_db(tworing,17.1352).  % [17.2055] 2004/3/9

hf_ring_exo(3,77.2743).  % [77.2717] [2004,5,9]
hf_ring_exo(4,25.0011).  % [24.9998] [2004,5,9]
hf_ring_exo(5,22.7663).  % [22.8592] [2004,5,9]
hf_ring_exo(6,23.808).  % [23.8077] [2004,5,9]
hf_ring_exo(other,19.601).

hf_ring_exo(ortho,3.16542).  % [3.07006] [2004,5,9]
hf_ring_exo(ortho-3,-7.25587).  % [-7.25444] [2004,5,9]
hf_ring_exo(ortho-_,0).
hf_ring_exo(ringdeform,19.2304).  % [19.2273] [2004,5,9]


hf_ring_deform(5,36.2641).  % [36.5326] [2004,5,9]
hf_ring_deform(6,48.187).  % [48.1863] [2004,5,9]




resonance2(roth,22.2591).  % [22.7139] 2004/3/9

resonance3(pp,12.4709).  % [13.0868] 2004/3/9

arom_ring_side(3,89.2055) :- !.  % [89.2781] 2004/3/9
arom_ring_side(4,30.5333) :- !.  % [30.6777] 2004/3/9
arom_ring_side(5,-3.99999) :- !.  % [-3.93155] 2004/3/9
arom_ring_side(6,3.95571) :- !.  % [3.96306] 2004/3/9
arom_ring_side(7,13.8824) :- !.  % [14.0146] 2004/2/6
arom_ring_side(other,4.56).
arom_ring_side(maromA,-0.019307).  % [-0.0340457] 2004/3/9
arom_ring_side(maromB,1.14471).  % [1.23226] 2004/3/9
arom_ring_side(db,-23.6863).  % [-22.3382] 2004/2/6
arom_ring_side(db5,45.5179).  % [46.5221] 2004/2/6

hf_rc(ortho_twist,93.8485).  % [93.7319] 2004/3/9
hf_rc(ortho,0).
hf_rc(meta,31.8791).  % [32.0315] 2004/3/9
hf_rc(para,65.0615).  % [65.1102] 2004/3/9
hf_rc(pp,-10.6374).  % [-10.6337] 2004/3/9
hf_rc(6,-9.67191).  % [-9.89086] 2004/3/9

%%hf_chiral_mult(pp,1).
hf_chiral_val(chiral,5.51326).  % [5.53842] 2004/3/9
hf_chiral_val(position,5.5).
hf_chiral_val(no2,4).
hf_chiral_val(oh-methyl,15).
hf_chiral_val(chiral5trans,26.0148).  % [25.224] 2004/3/9
hf_chiral_val(chiral6cis,18.3742).  % [16.3528] [2004,9,19]

hf_homarom(pp,-18.828).
```

```
hf_erc(5,7.27889).  % [8.56665] 2004/3/9
hf_erc(6,1.30906).  % [1.09055] 2004/3/9
hf_erc(7,11.4534).  % [11.131] 2004/3/9
hf_erc(8,0.548129).  % [0.207281] 2004/3/9

%hf_erc(a,5.54).
%hf_erc(b,-66.3).
%hf_erc(c,206.58).

%%hf_cis1_resonance(s,-0.381034).  % [-0.469702] 2004/3/9
hf_cis1_resonance(methyl,-6.80129).  % [-4.92593] [2004,4,29]
hf_cis1_resonance(f,-11.6107).  % [-11.5083] [2004,8,11]

hf_multi_ref(2,1.33823).  % [1.34782] 2002/1/9
hf_multi_ref(3,1.58636).  % [1.58609] 2001/11/29
hf_multi_ref(4,1.53321).  % [1.53321] 2001/11/29
hf_multi_ref(const,0.1).

hf_sigma_factor(0,1) . %% 66.9087).  % [61.9181] 2001/11/20
hf_sigma_factor(1,0.57943).  % [0.627737] 2002/1/9
hf_sigma_factor(2,0.395021).  % [0.401357] 2002/1/9
hf_sigma_factor(3,0.654941).  % [0.685741] 2001/11/29

hf_multi_charge(2,103.06).  % [115.557] 2002/1/9
hf_multi_charge(3,180.033).  % [179.3] 2001/11/29
hf_multi_charge(4,170.106).  % [169.546] 2001/11/29







hf_bridge(556,16.8015).  % [15.9033] 2004/3/9
hf_bridge(666,38.6746).  % [38.677] 2004/3/9
hf_bridge(567,28.9734).  % [28.8302] 2004/3/9
hf_bridge(578,-42.9828).  % [-42.8138] 2004/3/9
hf_bridge(668,-7.6174).  % [-7.4454] 2004/3/9
hf_bridge(778,-17.5677).  % [-17.4007] 2004/3/9
hf_bridge(888,-13.7023).  % [-13.5211] 2004/3/9
hf_bridge(677,-8.70504).  % [-8.78653] 2004/3/9

hf_bridge(5553,9.80447).  % [10.5532] 2004/3/9
hf_bridge(adamantane,17.6744).  % [17.8551] 2004/3/9
hf_bridge(protoadamantane,17.1183).  % [17.5064] 2004/3/9
hf_bridge(tetrane,52.8163).  % [53.5999] 2004/3/9
hf_bridge(cubane,3.78306).  % [5.08537] 2004/3/9
hf_bridge(bullvalene,-73.5292).  % [-73.5766] 2004/3/9
hf_bridge(bridgeane,34.777).  % [35.7171] 2004/3/9
hf_bridge(diadamantane,26.3004).  % [26.7009] 2004/3/9
hf_bridge(hatane,44.2785).  % [45.0771] 2004/3/9
hf_bridge(housane,-91.8675).  % [199.525] 2004/3/9
hf_bridge(a33bridge,41.0045).  % [41.6845] 2004/3/9
hf_bridge(azulene,-132.254).  % [-132.254] [2004,5,17]

hf_bridgedb(5,28.0485).  % [29.0434] 2004/3/9
hf_bridgedb(6,2.8587).  % [2.95356] 2004/3/9
hf_bridgedb(b6,-36.6864).  % [-36.8256] 2004/3/9
hf_bridgedb(7,30).

ring_vertices_1ring(0,0).
ring_vertices_1ring(1,-0.777639).  % [-0.631593] 2004/3/9
ring_vertices_1ring(2,5.0927).  % [5.28791] 2004/3/9
ring_vertices_1ring(cl,2.73934).  % [2.76993] [2004,9,19]
ring_vertices_1ring(br,-2.79614).  % [-2.7894] [2004,9,19]
ring_vertices_1ring(i,0).  % [7] [2004,7,22]
ring_vertices_1ring(f,-7.03189).  % [-4.19364] [2004,9,19]
ring_vertices_1ring(ff,27.73).  % [22.0679] [2004,9,19]
ring_vertices_1ring(ff3,34.4462).  % [28.8012] [2004,9,19]


ring_vertices_2ring(0,0).
ring_vertices_2ring(1,6.09436).  % [6.16001] 2004/3/9
ring_vertices_2ring(bridge,-2.30254).  % [-2.01174] 2004/3/9
ring_vertices_2ring(bridge2,27.0211).  % [27.0195] 2004/1/29
ring_vertices_2ring(methyl,1).
ring_vertices_2ring(f,0.0701933).  % [0.130209] [2004,9,19]


ring_vertices_3ring(0,0).
ring_vertices_3ring(1,-4.22699).  % [-3.41371] 2004/3/9
ring_vertices_3ring(ad-0,0).
ring_vertices_3ring(ad-1,-9.58532).  % [-9.5412] 2004/3/9
ring_vertices_3ring(spoke,-5.60694).  % [-5.71879] 2004/3/9
ring_vertices_3ring(flat,-7.07715).  % [-7.74242] 2004/3/9

ring_vertices_4ring(ad-0,0).
ring_vertices_4ring(ad-1,8.28913).  % [8.34161] 2004/3/9
```

```
hf_ring_side(side,1.01958).  % [1.10165] 2004/3/9
hf_ring_side(side5,-1.10748).  % [-0.795676] 2004/3/9
hf_ring_side(side6,-0.243663).  % [-0.208041] 2004/3/9

hf_ring_side3(3,11.0524).  % [11.1489] 2004/3/9
hf_ring_side3(4,3.95313).  % [4.05519] 2004/3/9
hf_ring_side3(5,0.492026).  % [0.585548] 2004/3/9
hf_ring_side3(6,1.81556).  % [1.8297] 2004/3/9
hf_ring_side3(7,-3.65254).  % [-3.57995] 2004/3/9
hf_ring_side3(8,-7.41585).  % [-7.34033] 2004/3/9

hf_ring_side3(a,-48.1781).
hf_ring_side3(b,75.6505).

oa_contrib(part,13.3836).  % [13.3833] 2003/7/22
brg_contrib(part,18.5708).  % [18.8396] 2003/7/22
buried_contrib(part,2.72901).  % [16.0499] 2003/7/22
bent_contrib(part,-53.7653).  % [-51.9023] 2003/7/22
res_contrib(part,-15.2842).  % [-15.6912] 2003/7/22
helical(part,33).



hf_connect_co2h(methyl,-25).
hf_connect_co2h(ethylenic,-50).
hf_connect_co2h(aromatic,-75).
hf_connect_co2h(_,0).

hf_connect_oh(methyl,63.6184).  % [63.8688] 2003/12/11
hf_connect_oh(oh,173.771).  % [63.8688] 2003/12/11
hf_connect_oh(ethylenic,41.734).  % [41.734] 2004/1/15
hf_connect_oh(_,63.6184).  % [63.8688] 2003/12/11


hf_branch_methyl(a,-4.23412).  % [-4.23698] 2004/3/9
hf_branch_methyl(b,35.5284).  % [35.524] 2004/3/9

hf_branch_methyl(methyl-3,0).
hf_branch_methyl(methyl-4,0).
hf_branch_methyl(oh-3,-12.8297).  % [10] 2003/12/11
hf_branch_methyl(oh-4,-16.1923).  % [10] 2003/12/11

hf_branch_oethylenic(pp,-8.57747).  % [-7.36389] [2004,4,29]

hf_branch_nethylenic(pp,8.68587).  % [8.68549] 2003/12/1
```

```
hf_branch_sethylenic(pp,8.68587).  % [8.68549] 2003/12/1

hf_steric_oethylenic(pp,13.2117).  % [7.62951] [2004,4,29]

hf_steric_nethylenic(pp,57.9297).  % [57.9195] 2003/12/1
hf_steric_sethylenic(pp,57.9297).  % [57.9195] 2003/12/1


hf_resonance(aromatic,21.7091).  % [21.7496] 2004/3/9
hf_resonance(acetylenic,0).  % [33.8477] 2004/3/9
hf_resonance(ethylenic,64.2616).  % [65.1357] [2004,4,29]
hf_resonance(oethylenic,51.8327).  % [15.6442] [2004,4,29]
hf_resonance(nethylenic,31.4051).  % [31.4231] 2003/12/1
hf_resonance(sethylenic,31.4051).  % [31.4231] 2003/12/1
hf_resonance(_,-40025).

hf_ring_exo_oethylenic(3,60.6425).  % [60.6419] [2004,5,9]
hf_ring_exo_oethylenic(4,-26.8695).  % [-26.8666] [2004,5,9]
hf_ring_exo_oethylenic(5,-18.2225).  % [-18.2208] [2004,5,9]
hf_ring_exo_oethylenic(6,-8.96039).  % [-8.96025] [2004,5,9]
hf_ring_exo_oethylenic(7,-27.1338).  % [-26.8482] [2004,5,9]
hf_ring_exo_oethylenic(8,-43.8089).  % [-43.8082] [2004,5,9]
hf_ring_exo_oethylenic(9,-39.3395).  % [-39.3387] [2004,5,9]
hf_ring_exo_oethylenic(10,-46.2684).  % [-46.2677] [2004,5,9]
hf_ring_exo_oethylenic(11,-44.9439).  % [-44.9432] [2004,5,9]
hf_ring_exo_oethylenic(12,-45.707).  % [-45.7063] [2004,5,9]
hf_ring_exo_oethylenic(15,-12.5441).  % [0] 2004/1/21
hf_ring_exo_oethylenic(17,-37.4991).  % [0] 2004/1/21

hf_exo_sigma(pp,6).

hf_hbonding(pp,-11.7767).  % [-10.8665] 2004/2/6

hf_dipole(oethylenic,24.28).  % [12.8631] [2004,4,29]

hf_steric_ring(pp,0.524269).  % [0.524232] [2004,5,9]
hf_steric_ring(3,21.9655).  % [21.9624] [2004,5,9]
hf_steric_ring(4,13.41). %half way between 3 and 5
hf_steric_ring(5,6.54219).  % [6.67269] [2004,5,9]
hf_steric_ring(6,12.1671).  % [12.1673] [2004,5,9]
hf_steric_ring(N,Val):-
number(N),
N > 6,
hf_steric_ring(6,Val).
```

```
/****
hf_relsz(3,0.869486) :- !.  % [0.192009] [2004,4,29]
hf_relsz(4,0.25) :- !.
hf_relsz(5,0.294506) :- !.  % [0.13785] [2004,4,29]
hf_relsz(6,0.815003) :- !.  % [0.225018] [2004,4,29]
hf_relsz(7,0.32) :- !.
hf_relsz(_,0) :- !.
***/

%hf_halo(f,0.580624).  % [0.207018] [2004,7,30]
hf_halo(f2temp,-19.4956).  % [-19.7962] [2004,8,11]
hf_halo(f332temp,15.1096).  % [14.7067] [2004,8,11]
hf_halo(f310temp,-34.7539).  % [-34.7853] [2004,8,11]
hf_halo(f310ringtemp,-46.2705).  % [-34.7853] [2004,8,11]
hf_halo(f3ethytemp,16.4884).  % [15.8181] [2004,8,11]
hf_halo(f3aromtemp,-4.95713).  % [-5.24736] [2004,8,11]
hf_halo(f2aromtemp,-22.3653).  % [-22.8936] [2004,8,11]
hf_halo(fethytemp,13.9461).  % [13.3556] [2004,8,11]

hf_halo(f-a,-6.4444).  % [-7.10747] [2004,9,19]
hf_halo(f-b,-4.35932).  % [0.27118] [2004,9,19]
hf_halo(cl-a,7.47735).  % [7.49664] [2004,9,19]
hf_halo(cl-b,-5.07362).  % [-5.1401] [2004,9,19]
hf_halo(br-a,0.379336).  % [0.509887] [2004,9,19]
hf_halo(br-b,12.4337).  % [11.9312] [2004,9,19]
hf_halo(i-a,24.5014).  % [24.4808] [2004,9,19]
hf_halo(i-b,-42.8356).  % [-42.7374] [2004,9,19]

hf_halo(cl-f,4.24612).  % [4.03925] [2004,9,19]
hf_halo(br-cl,5.3726).  % [5.44912] [2004,9,19]
hf_halo(br-f,1.27024).  % [1.1412] [2004,9,19]
hf_halo(f-i,8.37161).  % [8.15535] [2004,9,19]
hf_halo(cl-i,0).
hf_halo(br-i,0).

hf_halo(f-2,2.20834).  % [-0.160563] [2004,8,27]
hf_halo(f-3,-6.18326).  % [-14.1608] [2004,8,27]
hf_halo(f-4,-9.60171).  % [-17.8236] [2004,8,27]
hf_halo(cl-2,13.163).  % [11.7433] [2004,8,27]
hf_halo(cl-3,17.1716).  % [11.7433] [2004,8,27]
hf_halo(cl-4,29.7412).  % [11.7433] [2004,8,27]
hf_halo(br-2,12.6707).  % [13.4419] [2004,8,27]
hf_halo(br-3,23.296).  % [13.4419] [2004,8,27]
hf_halo(br-4,22.7055).  % [13.4419] [2004,8,27]
hf_halo(i-2,14.9798).  % [25.4632] [2004,8,27]
hf_halo(i-3,32.5325).  % [25.4632] [2004,8,27]
hf_halo(i-4,60.3095).  % [25.4632] [2004,8,27]

hf_halo(ringendo,-24.6554).  % [-25.0236] [2004,9,19]

hf_halo(per-1,-21.746).  % [-22.0368] [2004,9,19]
hf_halo(per-2,-0.201669).  % [-1.43299] [2004,9,19]
hf_halo(per-3,36.2017).  % [34.442] [2004,9,19]

hf_halo_xi(_-0,0).
hf_halo_xi(f-1,-16.6814).  % [-14.3884] [2004,9,19]
hf_halo_xi(cl-1,-1.7291).  % [-1.70757] [2004,9,19]
hf_halo_xi(br-1,-5.73919).  % [-5.74464] [2004,9,19]
hf_halo_xi(i-1,-2.65428).  % [-2.65441] [2004,9,19]
hf_halo_xi(f-2,-15.7148).  % [-17.0724] [2004,9,19]
hf_halo_xi(cl-2,-4.4341).  % [-4.41394] [2004,9,19]
hf_halo_xi(br-2,-17.1457).  % [-16.6787] [2004,9,19]
hf_halo_xi(i-2,0).
hf_halo_xi(clf-2,-13.5431).  % [-14.6977] [2004,9,19]
hf_halo_xi(clf-3,27.0086).  % [26.5684] [2004,9,19]
hf_halo_xi(brf-3,-15.3896).  % [-16.5755] [2004,9,19]
hf_halo_xi(fi-3,-1.98018).  % [-2.96178] [2004,9,19]
hf_halo_xi(f-3,-24.3379).  % [-25.2198] [2004,9,19]
hf_halo_xi(cl-3,5.62038).  % [5.66367] [2004,9,19]
hf_halo_xi(br-3,0).
hf_halo_xi(i-3,0).

hf_halo_xi(f-1-f-1,-7).
hf_halo_xi(cl-1-cl-1,-2.70979).  % [-4.36917] [2004,9,7]
hf_halo_xi(br-1-br-1,-6.58007).  % [-6.55895] [2004,9,7]
hf_halo_xi(i-1-i-1,-6.19635).  % [-6.59742] [2004,9,7]

hf_halo_xi(f-2-f-1,-32.449).  % [-6.24115] [2004,9,3]

hf_halo_xi(f-f,-3.88949).  % [-3.82327] [2004,9,19]
hf_halo_xi(cl-f,1.85674).  % [1.55895] [2004,9,19]
hf_halo_xi(br-f,-1.22865).  % [-0.996293] [2004,9,19]
hf_halo_xi(f-i,-1.46654).  % [-1.57734] [2004,9,19]
hf_halo_xi(cl-cl,1.5597).  % [1.58067] [2004,9,19]
hf_halo_xi(br-cl,15.1055).  % [14.4354] [2004,9,19]
hf_halo_xi(cl-i,-7).
hf_halo_xi(br-br,-7).
hf_halo_xi(br-i,-7).
hf_halo_xi(i-i,-7).
hf_halo_xi(f3-e,1.80871).  % [0.613776] [2004,9,19]
hf_halo_xi(f3-arom,-16.5459).  % [-17.4856] [2004,9,19]

hf_halo_ethy1(per-2,-54.6922).  % [-50.1494] [2004,9,19]
hf_halo_ethy1(per-3,100).

hf_halo_ethy1(f-f,-1.43122).  % [2.31213] [2004,9,19]
hf_halo_ethy1(cl-cl,2.26072).  % [2.17074] [2004,9,19]
hf_halo_ethy1(br-br,0).
hf_halo_ethy1(i-i,0).
hf_halo_ethy1(cl-f,0).
hf_halo_ethy1(br-f,0).
hf_halo_ethy1(f-i,0).
hf_halo_ethy1(br-cl,0).
hf_halo_ethy1(cl-i,0).
hf_halo_ethy1(br-i,0).

hf_halo_ethy2(f-f,24.6575).  % [23.6007] [2004,9,19]
hf_halo_ethy2(cl-cl,7.46628).  % [7.37682] [2004,9,19]
hf_halo_ethy2(br-br,0).
hf_halo_ethy2(i-i,0).
hf_halo_ethy2(cl-f,0).
hf_halo_ethy2(br-f,0).
hf_halo_ethy2(f-i,0).
hf_halo_ethy2(br-cl,0).
hf_halo_ethy2(cl-i,0).
hf_halo_ethy2(br-i,0).

hf_halo_ethy3(f-f,31.9963).  % [33.015] [2004,9,12]
hf_halo_ethy3(cl-cl,0).
hf_halo_ethy3(br-br,0).
hf_halo_ethy3(i-i,0).
hf_halo_ethy3(cl-f,0).
hf_halo_ethy3(br-f,0).
hf_halo_ethy3(f-i,0).
hf_halo_ethy3(br-cl,0).
hf_halo_ethy3(cl-i,0).
hf_halo_ethy3(br-i,0).
```