SimSoup Project

The SimSoup Guide

Conceptual Model, Logical Model, and User Manual

Version 1.2
10 October 2013

Chris Gordon-Smith
www.simsoup.info
c.gordonsmith@gmail.com

Version History		
Version	**Date**	**Change Description**
1.0	13 January 2013	First issue
1.1	18 August 2013	Added User Manual
1.2	10 October 2013	Added 'Could Life Have Always Existed?' and Acknowledgements sections

Contents

1 Introduction

This document is the SimSoup Guide. It explains the SimSoup Artificial Chemistry Simulator (version 0.61). It includes the SimSoup Conceptual and Logical Models, and the User Manual for running the simulator.

1.1 Motivation For SimSoup

1.1.1 Inheritance at the Origin of Life

There is currently no complete and generally accepted explanation of the Origin of Life. Explaining this entails explaining the Origin of Evolution. Since Darwin, the process of evolution by which new species of life-form originate has been well understood. By contrast, it is not known how the process *started*. A key question that must be answered is: *How did entities capable of maintaining and transferring inherited information arise?* There are two main views:

- *Genetic View:* Template replicating structures such as RNA, DNA, or mineral crystals were crucial for the Origin of Life, and have from the outset been the carriers of the inherited information that makes evolution possible
- *Metabolic View:* The first living entities were metabolic systems, and they evolved by exploring the possibilities for new kinds of metabolic network.

Much attention has focused on the RNA World and other genetic theories, but a full explanation for the origin of accurately replicating template molecules remains elusive. The main problem is the extreme improbability of such molecules arising through a purely random process.

Metabolic View theories take a simpler starting point. In their 'heterotrophic' variants they require sufficient concentrations of preformed organic molecules such as amino acids or lipids. In 'autotrophic' variants the starting point is even simpler; small inorganic molecules are all that is required.

Appealing to simplicity does not relieve Metabolic View theories of the obligation to explain how a metabolic system with no template replicator can maintain and transfer inherited information. Finding an explanation is the motivation for the SimSoup project.[1]

With its metabolism oriented focus, the SimSoup project does not take a strong view on whether the first evolving systems were heterotrophic or autotrophic,[2] or on the environment in (or on) which they existed.

[1] For details of progress of the SimSoup project, see (Gordon-Smith, 2005, 2007, 2009a,b, 2011, 2013).

[2] "You can have a soup of anything": Günter Wächtershäuser, during a conversation at the 'Conditions for the emergence of life' conference at the Royal Society in 2006.

1.1.2 Biological and Chemical Computing

The simple memory mechanisms being investigated by SimSoup are also relevant in this newer field. For example, it may be possible to build a 'real chemistry' memory system based on SimSoup concepts, and to use this as the basis of an inheritance mechanism for the evolution of artificial systems.

1.2 Open Ended Network Discovery

Molecular structure and the way molecules react are both determined according to physical laws. The number of possible molecular species is vast, as is the number of possible reactions. There is an effectively infinite network of possible molecular species and reactions that is determined for all time by the laws of physics.

In any particular chemical environment, only a few of these possible species and reactions are realised. When new molecular species arise, new reactions become possible. These reactions may in turn give rise to further molecular species as products.

When simulating chemistry it is clearly not possible to specify this vast network as initial conditions. An alternative in which a finite set of molecular species and reactions is specified is possible, but this has the drawback that the reaction network is then finite and closed, limiting the scope for novelty.

The approach taken by SimSoup is the same as that taken in real chemical environments. The laws governing the molecular structures and interactions that are possible are fixed, but the molecular structures and interactions are found as the simulation runs in a process of *open-ended network discovery*.

1.3 Could Life Have Always Existed?

Today it is accepted that there was a time when there was no life, and that life must therefore have had an *origin*. This has not always been the case. In the nineteenth century, Hermann Richter put forward the idea that life has always existed in the universe, propagating itself from one place to another by means of 'cozmozoa' (germs of the cosmos). In this theory, life has existed and will exist for all eternity, and so there is no need for an explanation of its origin. Lord Kelvin and Herman von Helmholtz also took the view that life on Earth arrived from elsewhere in the universe.

The idea that life on Earth had an extra-terrestrial origin has a very long history. It can be traced back to the ancient Greek philosopher Anaxagoras. He claimed that the universe is made of an infinite number of spermata (seeds) that give rise to life forms on reaching the Earth. Anaxagoras coined the term Panspermia, meaning 'seeds everywhere', for his proposal.

This historical note highlights that the fact that life exists does not *necessarily* imply that it had an origin.

1.4 Document Overview

Section 2 presents a Conceptual Model of chemistry. This model incorporates the key aspects of 'real' chemistry on which SimSoup is based. Although SimSoup is an artificial chemistry project, the intention is that it should be as realistic as possible.

Section 3 presents the SimSoup Logical Model. This is a specification of:

- The major components of the simulator: the Chemistry and the Reactor
- The types of object used to represent molecular structure and chemical process
- A description of the overall operation of the simulator, including the open ended way in which the vast (artificial) chemical network is *discovered* as the simulation runs.

Section 4 presents the User Manual for SimSoup.

2 Conceptual Model

This section presents a conceptual model of chemistry that forms the basis of the SimSoup Logical Model described in Section 3. This abstraction has been developed taking account of the SimSoup project's aim to explain the Origin of Evolution in terms of metabolic systems. Chemistry is seen as a framework determining the structural properties of molecules, and the nature of the chemical processes in which they can participate.

Although the conceptual model is an abstraction, it is intended to be realistic; much of the material that follows will be familiar to physical chemists. See for example Atkins and Paula (2006), Atkins (2001), and Pauling (1960).

2.1 Molecular Structure

2.1.1 Atoms and Elements

The basic structural unit in chemistry is the *atom*. Atoms are of different types. These types are called *elements*. An atom has a *mass* and a number of *electrons* that orbit a *nucleus*. Some of these are *valence electrons* that can participate in bonds with other atoms.

2.1.2 Bonds and Atomic Shells

Atoms can bond in different ways. Common mechanisms include *ionic bonding*, *covalent bonding*, and *hydrogen bonding*.

In covalent bonding, atoms 'share' valence electrons. The stability of a bond is due to the electron sharing configuration having a lower energy than the non-sharing (unbonded) configuration.

This can be understood as follows. Electrons orbit an atomic nucleus in one or more *shells*. Electron shells each have a *shell capacity*; this is the maximum number of electrons that can exist in the shell. Atoms 'prefer' to have full electron shells, because such configurations have lower energy than other configurations. Electron sharing involves a pair of electrons being 'partly in' a shell belonging to one atom, and 'partly in' a shell belonging to the other, such that the shells of both atoms are full, or closer to being full.

For example, an Oxygen atom and a Hydrogen atom can bond as follows. Hydrogen has a single electron shell with a shell capacity of two. In the unbonded state it has a single valence electron in this shell. Oxygen has six valence electrons in a shell that has a capacity of eight.[1] The two atoms can share a pair of electrons. In this configuration the Hydrogen atom has two electrons and its shell is full, and the Oxygen shell has seven electrons.

Further, the Oxygen atom can share a second electron with another Hydrogen atom. The resulting structure, water, is stable because the electron shells of all three atoms are full.

Each covalent bond has a *bond order*. This is the number of electron pairs being shared. In a double bond two electron pairs are shared; in a triple bond three electron pairs are shared.

2.1.3 Bond Enthalpy

There is an amount of energy, the *bond enthalpy*, associated with each bond.[2] Two atoms cannot be separated unless at least this amount of energy is supplied. When two atoms join the energy is released, usually as heat. The enthalpy of a bond depends on the types of the two atoms at either end. However, it can also vary depending on the particular molecular species. For example, the mean molar bond enthalpy[3] for a single order $C-C$ bond is 348 $kJmol^{-1}$. By contrast, the single order $C-C$ bond in ethane (H_3C-CH_3) has molar bond enthalpy 368 $kJmol^{-1}$.

2.1.4 Molecules And Their Structure

A group of atoms bonded together into a single unit is called a *molecule*. Each molecule has a particular

[1] In addition, it has two electrons in an inner shell that is full.

[2] Chemists typically refer to molar bond enthalpies in units of kilojoules per mole ($kJmol^{-1}$). That is, kilojoules per N_A molecules each containing one bond of the kind under consideration, where N_A (Avogadro's number) = 6.0221413 x 10^{23}. In the SimSoup Conceptual Model, a bond enthalpy is *for an individual molecule*; this is equivalent to setting $N_A = 1$. In this document, where the intention is to refer to bond enthalpy per mole, then this is made explicit by using the term *molar bond enthalpy*.

[3] The mean molar bond enthalpy is the average of molar bond enthalpies over a related series of compounds. See Atkins (2001), page 65.

structure. Two molecules including the same number of atoms of each element bonded together into identical structures are both members of the same *molecular species*.

2.1.5 Ions

In chemistry, molecules are usually considered to be electrically neutral. In many situations, chemistry involves charged particles; such particles are called *ions*. An ion may be produced as a result of adding or removing electrons to or from an atom or molecule.

2.1.6 Atomisation Enthalpy Of Molecules And Ions

The *atomisation enthalpy* of a molecule or ion is the energy required to fully break all the bonds between its atoms.

2.2 Chemical Processes

What is a chemical process? How can it be represented as a network? How fast does the process go? The following subsections describe the key features.

2.2.1 Components Of A Chemical Process

2.2.1.1 Elementary Reactions

Most chemical processes occur in a sequence of steps called *elementary reactions*. These are the basic units or components of a chemical process. An elementary reaction involves only a small number of molecules or ions. For example:

$$NO_2 + NO_3 \rightarrow N_2O_5 \quad (2.1)$$

$$O_3 \rightarrow O_2 + O \quad (2.2)$$

$$H + Br_2 \rightarrow HBr + Br \quad (2.3)$$

$$\textit{cyclo}\text{-}C_3H_6 \rightarrow CH_3CH{=}CH_2 \quad (2.4)$$

The left hand side of each reaction shows the *reactant(s)* that are converted to the *product(s)* on the right.

In reaction 2.1, a molecule of nitrogen dioxide (NO_2) and a molecule of nitrogen trioxide (NO_3) combine to form a molecule of dinitrogen pentoxide (N_2O_5).

In reaction 2.2, a triatomic molecule of ozone (O_3) consisting of three oxygen atoms splits to become diatomic molecular oxygen (O_2) plus a free oxygen atom.

In reaction 2.3, an atom of hydrogen (H) and a diatomic molecule of bromine (Br_2) come together. The hydrogen atom bonds to one of the bromine atoms, and the other bromine atom is released.

In reaction 2.4, a molecule of cyclopropane *isomerises* to a molecule of propene. This means that its atomic composition remains the same, but the molecule changes from a cyclic structure to a chain structure including a double bond.

2.2.1.2 Molecularity

The *molecularity* of an elementary reaction is the number of reactant molecules (or ions). *Unimolecular* reactions have one reactant, *bimolecular* reactions have two. In either case, there may be one or more product molecules or ions (usually only a few).

Reactions 2.1 and 2.3 are examples of bimolecular reactions. Reactions 2.2 and 2.4 are examples of unimolecular reactions.

Unimolecular and bimolecular elementary reactions have the following general forms:

$$R \rightarrow P_1 + P_2 + ...P_n \quad \text{Unimolecular} \quad (2.5)$$

$$R_1 + R_2 \rightarrow P_1 + P_2 + ...P_n \quad \text{Bimolecular} \quad (2.6)$$

Elementary reactions involving three reactants are called *termolecular*. Termolecular reactions in solutions or gas mixtures are rare, due to the improbability of collisions involving three molecules.

2.2.1.3 Combining Elementary Reactions

Elementary reactions often combine to form the steps of an overall reaction process. For example, ozone can decompose with the overall reaction:

$$2O_3 \rightarrow 3O_2 \quad (2.7)$$

The elementary reactions involved include:

$$O_3 \rightarrow O_2 + O \quad (2.8)$$

$$O_2 + O \rightarrow O_3 \quad (2.9)$$

$$O + O_3 \rightarrow O_2 + O_2 \quad (2.10)$$

In Reaction 2.8, a molecule of ozone splits to produce a diatomic oxygen molecule, and an atom of oxygen. Reaction 2.9 is the reverse of reaction 2.8; some of the ozone is replaced by the atomic oxygen and diatomic oxygen recombining. However, other atoms of oxygen react (Reaction 2.10) with ozone such that the ozone loses one oxygen atom, and the two oxygen atoms combine to produce diatomic oxygen.

The atomic oxygen is an *intermediate*. It does not appear in Reaction 2.7, the overall reaction, but is necessary for the final step, elementary reaction 2.10.

The following shows another example, in this case the decomposition of N_2O_5. The overall reaction is:

$$2N_2O_5 \rightarrow 4NO_2 + O_2 \quad (2.11)$$

The elementary reactions are:

$$N_2O_5 \rightarrow NO_2 + NO_3 \quad (2.12)$$

$$NO_2 + NO_3 \rightarrow N_2O_5 \quad (2.13)$$

$$NO_2 + NO_3 \rightarrow NO_2 + O_2 + NO \quad (2.14)$$

$$NO + N_2O_5 \rightarrow NO_2 + NO_2 + NO_2 \quad (2.15)$$

2.2.2 Representing Chemical Processes As Networks

Figure 2.1 shows the network structure for the ozone decomposition process of reactions 2.8 to 2.10. The network has two kinds of node; those along the bottom correspond to the reactants and products of the elementary reactions. The other nodes correspond to the elementary reactions themselves. The top left node represents elementary reaction 2.8. The arrow to it from O_3 represents ozone being consumed. The arrows leaving it represent atomic oxygen (O) and molecular oxygen (O_2) being released as products.

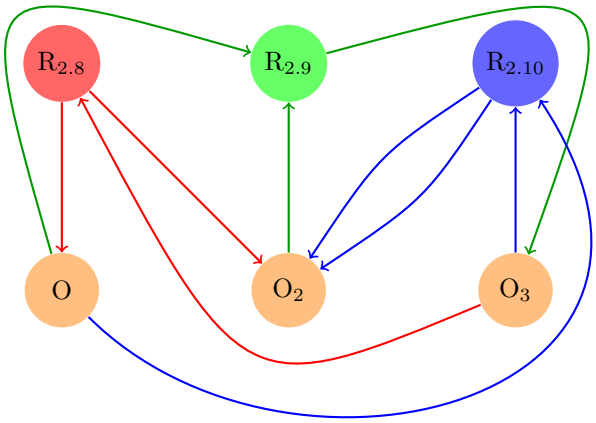

Figure 2.1: Elementary reactions for ozone decomposition

Figure 2.2 shows another example, in this case for the decomposition of N_2O_5 shown in reactions 2.12 to 2.15.

Reaction networks are not often drawn in this way. Diagrams are more often seen in which reactions are shown as lines between molecular species. However, while these diagrams are more easily understood, they often omit details.[4]

In order to *fully* specify a reaction network it must be possible to define the molecular species, and an arbitrary number of associations (elementary reactions) between them. Each of the associations has one or two 'inputs' (three in the case of rarer termolecular reactions), and an arbitrary number of 'outputs' (usually only a few).

The notation of Figures 2.1 and 2.2 meets this requirement. In principle, a reaction network of any complexity can be fully represented. Although such networks are not often drawn in full, it is not possible to

[4]Try drawing Figure 2.2 without using 'reaction' nodes!

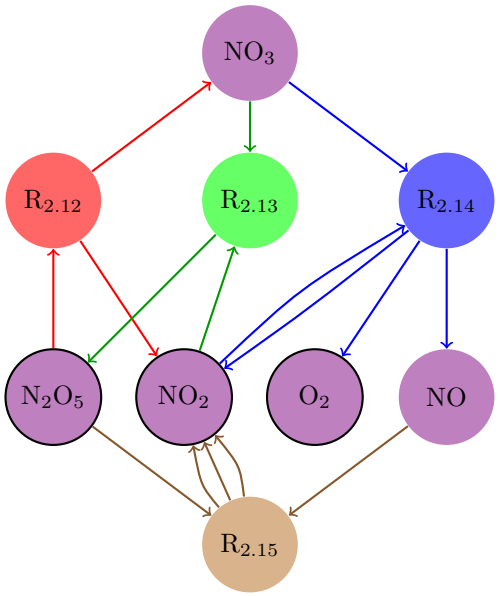

Figure 2.2: Elementary reactions for N_2O_5 decomposition

model the dynamic behaviour of a general reaction network without a full description of this kind. In some cases a simplified graphical notation *is* possible; this is discussed in section 3.2.4.

Mathematicians call networks of the kind shown in Figures 2.1 and 2.2 *directed bipartite graphs*.[5]

2.2.3 Reaction Rates

The rate of a chemical process depends on the rates of the individual elementary reactions involved. This section describes the factors determining the rate of an elementary reaction.

2.2.3.1 Kinetic and Potential Energy

In any elementary reaction, a set of reactants is transformed into a different set of products. Although mass and energy must both be conserved, there is usually a transformation of energy between two forms, kinetic and potential.

Kinetic energy is energy of motion. In chemical reactions it is usually thermal motion, but can also be radiative (for example from ultra-violet light).

[5]A bipartite graph has nodes that can be divided into two disjoint sets U and V such that every edge connects a node in U to one in V. In a directed bipartite graph, each edge has a direction.

Alternatively, a chemical network can be represented by a *directed hypergraph*; a hypergraph is a generalisation of a graph in which a 'hyperedge' can connect any number of nodes. In a directed hypergraph, hyperedges connect 'head' nodes to 'tail' nodes. The nodes would represent molecular species, and the hyperedges would represent elementary reactions.

Potential energy is energy associated with a system's configuration, and is measured by reference to some alternative configuration of the system. For example, if a spring starts from its unstretched state and is then stretched, the work done in stretching it is converted to potential energy. This potential energy can be released if the spring is allowed to return to its unstretched state. Similarly, if two oppositely charged particles are moved apart from an initial position, the system acquires potential energy that can be released if the particles move back to their initial configuration.

A molecule or ion of a particular species has a defined potential energy. When an elementary reaction takes place, the energy released or absorbed is the difference between the potential energy of the reactants and the products.

Elementary reactions that release heat energy by transforming potential energy to kinetic energy are called *exothermic*. Elementary reactions that transform heat energy to potential energy are called *endothermic*.

2.2.3.2 Entropy

The concept of entropy has been summarised by Frank Lambert as follows:[6]

> Entropy is not disorder. Entropy is not a measure of disorder or chaos. Entropy is not a driving force. Energy's diffusion, dissipation, or dispersion in a final state compared to an initial state is the driving force in chemistry. Entropy is the index of that dispersal within a system and between the system and its surroundings

2.2.3.3 Reaction Profile

Any elementary reaction has a *reaction profile* that shows how potential energy varies as the reaction progresses.

Figure 2.3 shows an example reaction profile. The reaction coordinate represents how far the elementary reaction has progressed. On the left hand side of the figure, the reaction coordinate represents a configuration in which reactants are about to interact. On the right hand side of the diagram, the reaction coordinate represents a configuration in which products have just been formed.

The figure shows that the reactants initially have a potential energy V_r, and that as the reaction progresses the potential energy increases until it reaches a maximum value, after which the potential energy decreases until it reaches a value V_p associated with the products.

[6]From "Disorder - A Cracked Crutch For Supporting Entropy Discussions", available at www.entropysite.com. Originally published in: J. Chem. Educ. 2002 79 187-192.

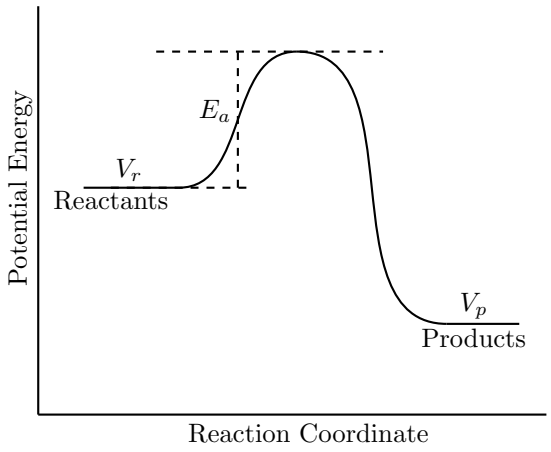

Figure 2.3: A Reaction Profile

The configuration that corresponds to the region close to the maximum is called the *activated complex*. The difference between the initial potential energy of the reactants, V_r, and the maximum associated with the activated complex is known as the *activation energy*, E_a. The activation energy is the minimum kinetic energy that the reactants must have in order for the reaction to proceed.

2.2.3.4 The Rate Constant

The rate of occurrence of an elementary reaction depends on its *rate constant*, k, and the concentration(s) of the reactant(s).[7]

In a unimolecular reaction with reactant A, the total rate V of the reaction is:

$$Rate = k[\mathrm{A}]$$

where [A] is the concentration of species A. In a bimolecular reaction with reactant species B and C, the total rate of the reaction is:

$$Rate = k[\mathrm{A}][\mathrm{B}].$$

2.2.3.5 The Arrhenius Parameters

Chemical reactions usually occur faster at higher temperatures. It is found experimentally that many elementary reactions have rate constants that vary with temperature according the Arrhenius equation:

$$k = Ae^{-E_a/RT}$$

where T is temperature and R is the gas constant.

The relationship holds for both unimolecular reactions and bimolecular reactions, and for reactions taking place both in gases and in solutions.

A and E_a are known as the *Arrhenius Parameters*.

[7]It will be seen below that although k is called the rate constant, it is *temperature dependent*.

2.2.3.6 Interpretation Of The Arrhenius Parameters

The Exponential Factor

The factor $e^{-E_a/RT}$ in the Arrhenius equation is called the *exponential factor*. We can interpret this factor as the fraction of potential reaction events in which sufficient kinetic energy is available to overcome the activation energy, E_a.

In the case of a reaction taking place between two molecules in the gas phase, $e^{-E_a/RT}$ is given by the Boltzmann distribution as the fraction of collisions with a kinetic energy in excess of E_a. This kinetic energy would be derived from the thermal motion of the two reactant molecules. In the case of a liquid phase reaction taking place in a solution, some or all of the kinetic energy would be derived from thermal energy of the solvent molecules.

The Pre-Exponential Factor

The *pre-exponential factor*, A, can be interpreted as the frequency of potential reaction events. In the case of a gas phase reaction between two colliding molecules, two factors affect this frequency:

- *Collision Event Frequency*: This is the frequency of events in which two molecules approach one another closely enough to react
- *Steric Factor*: In some cases, a close approach of two molecules is not sufficient for a reaction, they must also approach each each other with the right orientation. In such cases it is said that the reaction has a steric requirement.

In the liquid phase, reactants may approach one another as a result of diffusing through a solution. Migration through diffusion is much slower than the motion of molecules in a gas, and so encounters in a liquid may be rarer. However, the encounter may last longer due to the slower migration, and as a result of the *cage effect* in which the surrounding solvent molecules can hinder migration.

Activation Entropy

Consider a reversible unimolecular reaction:

$$\mathrm{X} \rightleftharpoons \mathrm{Y}$$

Suppose that the activation energy for the reverse reaction is greater than for the forward reaction (as is the case in Figure 2.3, in which the potential energy of the products is less than that of the reactants).

In this situation, the exponential factor $e^{-E_a/RT}$ will be larger for the forward reaction than for the reverse reaction. If the conditions determining the pre-exponential factor, A, are such that A has the same value for both the forward and reverse reactions, then the rate constant for the forward reaction will be greater than that for the reverse reaction.

This reflects the fact that it is easier for a molecule of X to acquire the activation energy for the forward reaction than for a molecule of Y to acquire the higher activation energy required for the reverse reaction.

It is not necessarily the case that a reversible reaction always prefers the direction that results in lower potential energy. For example, consider the following bimolecular reaction:

$$\mathrm{C} + \mathrm{D} \rightleftharpoons \mathrm{E} + \mathrm{F}$$

Suppose that there is a very stringent steric requirement for the forward reaction such that the C and D molecules must approach each other with a very specific orientation. Suppose also that there is no such stringent requirement for the reverse reaction. This would lead to a situation in which the pre-exponential factor for the forward reaction is very low by comparison with the reverse reaction. It follows that the rate constant for the reverse reaction can be higher than that for the forward reaction, even if the activation energy for the reverse reaction is higher.

This can be understood in terms of an *activation entropy* component in the pre-exponential part of the Arrhenius equation. In the bimolecular example above, the stringent steric requirement for the forward reaction means that the products can have a lower entropy than the reactants. This can outweigh the exponential factor $e^{-E_a/RT}$ that would favour the forward reaction.

Rate Constant Depends On Energy And Entropy Of Activation

We can conclude from the above that the rate constants of reactions between molecular species in chemical networks are determined by thermodynamic factors including the activation energy *and* activation entropy, and that the Arrhenius parameters are a reflection of the operation of these factors. A fuller description of this topic in terms of transition state theory is beyond the scope of this document.

3 The SimSoup Logical Model

This section presents the SimSoup Logical Model. Subsection 3.1 outlines two major components, the *Chemistry* and the *Reactor*. Subsections 3.2 and 3.3 give more details about these components.

Section 3.4 goes on to describe the overall operation of the simulation, including how the Chemistry and Reactor objects are setup, how chemical processes take place, and the open-ended way in which the chemical network is *discovered* as the simulation runs.

3.1 The Chemistry And The Reactor

The SimSoup Logical Model includes two major components called the Chemistry and the Reactor.

The Chemistry corresponds to the laws of chemistry. It determines what *can* happen, and consists of Atom Types, Bond Types, Molecule Types and Interaction Types.

The Reactor is where events *do* happen; it is the environment in which actual Molecules exist and interact. It is assumed to be 'well stirred', so that a particular Molecule is equally likely to interact with any other Molecule. It also has a Temperature and a Volume. In addition, the Reactor is a 'leaky' environment. The rate of leakage (which can be zero) determines the proportion of Molecules that leave the Reactor each timestep. These characteristics of the Reactor are shown in Table 3.1.

Reactor Characteristics	
Name	**Description**
T	Temperature in the Reactor
V	Volume of the Reactor
Reactor Leakage	Proportion of Molecules that are removed from the Reactor each timestep. Each Molecule is equally likely to leave

Table 3.1: Reactor Characteristics

3.2 Chemistry Objects

This section describes the types of SimSoup object that represent molecular structure and elementary reactions as described in Sections 2.1 and 2.2.

3.2.1 Atom Types

Atom Types in the Logical Model correspond to elements in the Conceptual Model. Table 3.2 describes the characteristics of an Atom Type.

Atom Type Characteristics	
Name	**Description**
$Name$	Name of the Atom Type
$Symbol$	Symbol for the Atom Type
M_{AType}	Mass of the Atom Type
$N_{ValElectrons}$	Number of valence electrons
C_{Shell}	Total shell capacity for the shell(s) containing valence electrons
$Colour$	Colour of the Atom (for display)

Table 3.2: Atom Type Characteristics

3.2.2 Bond Types

A SimSoup Atom of a particular Atom Type may bond to another Atom, which may be of the same Atom Type or of a different Atom Type.

The bonds are envisaged as covalent (eg rather than ionic). The maximum number of bonds that can be supported is $N_{PossBond} = C_{Shell} - N_{ValElec}$. In other words, the number of possible bonds for an Atom is the number of extra electrons required to fill its electron shell(s).

Table 3.3 shows the characteristics of a Bond Type. The Default Bond Enthalpy ($H_{BondDefault}$) for a Bond Type is the *usual* energy associated with Bonds of this type (ie the energy usually required to break them).

In some situations in SimSoup, the Enthalpy of a Bond depends not only on the Atom Types at either end of the Bond, but also on the configuration of other Atoms in the Molecule in the neighbourhood of the Bond. Particular configurations can strengthen or weaken the Bond (Bond Enthalpy increased or decreased). Such configurations are called Perturbation Configurations.

Bond Type Characteristics	
Name	**Description**
$AType_1$	First Atom Type
$AType_2$	Second Atom Type
$BondOrder$	Bond Order
$H_{BondDefault}$	Default Bond Enthalpy
$PertConfs$	Zero, one, or many Optional Perturbation Configurations (see Table 3.4)

Table 3.3: Bond Type Characteristics

A Bond Type need not have Perturbation Configurations, but can have many if required. Table 3.4 shows the characteristics of a Perturbation Configuration. It has a ΔH_{Bond} (the amount by which the Bond Enthalpy is increased or decreased), and one or more perturbing Atom Types, each with a specified location. In order for a Bond in a Molecule to be affected by a Perturbation Configuration, each Atom Type in the Perturbation Configuration must match an Atom in the Molecule.

Atoms that perturb a Bond in a Molecule are not bonded to either of the Bond's Atoms; they are just 'nearby' at the specified locations in the Molecule.

Note that X_{Offset} and Y_{Offset} are relative to Atom 1 in the Bond. Atom 2 is assumed to be at offset (1,0) from Atom 1. That is, one Bond length to the right of Atom 1.

Perturbation Configuration Characteristics	
Name	**Description**
ΔH_{Bond}	Amount by which Bond Enthalpy is increased or decreased
Followed by one or more groups of...	
AType	Atom Type that can perturb the Bond Type
X_{Offset}	X offset of the perturbing Atom from bond Atom 1
Y_{Offset}	Y offset of the perturbing Atom from bond Atom 1

Table 3.4: Pertubation Configuration Characteristics

3.2.3 Molecule Types

Molecule Types correspond to molecular species in the Conceptual Model. Electrical charge is not currently represented in the SimSoup Logical Model, and so there are no objects corresponding to ions. Table 3.5 shows the characteristics of Molecule Types.

Figure 3.1 shows some example SimSoup Molecule Types. Atoms are colour coded as indicated in the caption. The four Molecule Types at the top of the figure are analogous to some small 'real chemistry' molecular species. The other two Molecule Types in the figure are larger, and these examples have bonding sites. For example, in Figure 3.1e the Carbon Atom at the top right has three of its four valences unused. The lower Oxygen Atom has one unused valence and could bond to a Hydrogen Atom.

3.2.4 Interaction Types

Interaction Types correspond to 'platonic' types of elementary reaction. The use of the term 'platonic' here is to make the distinction between a *type of reaction*, and an *actual* reaction.

Each Interaction Type has one or two *Reactant* Molecule Types for Molecules consumed by Interactions, and one or more *Product* Molecule Types for Molecules produced.

The Logical Model includes Interaction Types of the following forms:[1]

- *Construction*: Two Reactant Molecules join to form a single Product Molecule
- *Transformation*: A Molecule rearranges into an *isomer*; a Molecule with the same atomic composition but different Structure
- *Fission*: A single Reactant Molecule splits to form two Product Molecules.

[1]The Logical Model may be extended in future to include Interaction Types with different numbers of Products. Transformations are not currently implemented in SimSoup.

Molecule Type Characteristics	
Name	**Description**
Identifier	Unique identitier for the Molecule Type
Name	Name (optional)
Structure	Molecules are two dimensional rigid structures built from Atoms. Except in the case of a Molecule consisting of a single Atom, the Atoms are bonded together such that they occupy fixed positions on a rectangular 'Board' (similar to a chess board). Bond angles are always 90° or 180°, and bond lengths are all equal. Other bond characteristics are as described in Section 3.2.2. Each square contains at most one Atom. Squares are identified with Row and Column numbers. For example, the square at the top left of the 'Board' containing a Molecule Type has Row Number = 1 and Column Number = 1. The Row and Column numbers relate to the Molecule Type spatially orientated as displayed on the SimSoup user interface
The Structure also determines the following for the Molecule Type...	
Mass	The sum of the Masses of all the Atoms
BondingSites	The number of Atoms in the Molecule Type whose electron shells are not full
Atomisation Enthalpy	The energy required to atomise the Molecule Type. That is, to fully break all of its Bonds

Table 3.5: Molecule Type Characteristics

Constructions are bimolecular reactions, Fissions and Transformation are unimolecular reactions.

Each of the above forms of Interaction Type involves either two or three Molecule Types. This makes it possible to represent interaction networks using diagrams in which nodes representing Molecule Types are joined by lines. As indicated in Section 2.2.2, this is not possible in more general cases involving more Products.

Figure 3.2 shows the three forms of Interaction Type using this alternative graphical convention. Note the annotations that identify the Interaction Types (C1, T1 and F1).

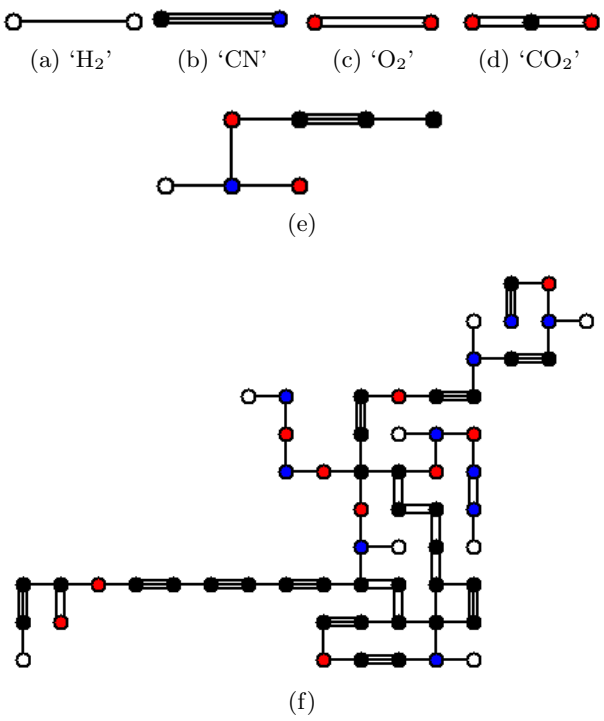

Figure 3.1: Example SimSoup Molecule Types. Atoms are colour coded as follows: Carbon - Black, Nitrogen - Blue, Oxygen - Red, Hydrogen - White.
a) Diatomic 'Hydrogen'. b) 'Cyanide'. c) Diatomic 'Oxygen'. d) 'Carbon Dioxide'. e) A Molecule with two bonding sites (the right hand Oxygen and Carbon Atoms). f) A large Molecule with (only) two bonding sites; the two Nitrogen Atoms in the centre.

Table 3.6 shows Interaction Type characteristics. The Activation Energies mentioned in the table are calculated as shown in Table 3.7. Table 3.8 defines the simulation parameters A_1, A_2, E_{aMin1} and E_{aMin2} used in Tables 3.6 and 3.7. T is described in Table 3.1.

A_1 and A_2 correspond to the Arrhenius pre-exponential factor in the forward and reverse directions. In the Conceptual Model these are different for each elementary reaction. Adopting one forward and one reverse factor here is a model simplification.

3.3 Reactor Objects

This section describes the types of SimSoup object that represent the contents of the Reactor and the actual processes occurring in it.

3.3.1 Molecules

A Molecule is an instance of a Molecule Type. Its characteristics are as defined by the Molecule Type.

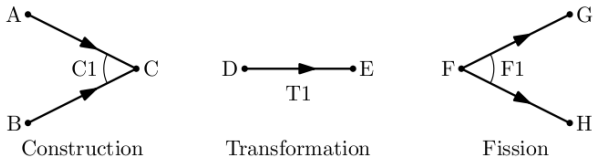

Figure 3.2: The three forms of Interaction Type. In Construction C1, Reactant Molecules of types A and B join to form a Product of type C. In Transformation T1, a Molecule of type D re-arranges to form a Molecule of type E. In Fission F1, a Molecule of type F splits into Molecules of types G and H.

Interaction Type Characteristics	
Name	**Description**
$Identifier$	Unique identifier for the Interaction Type
$Category$	Construction, Transformation or Fission
$\Delta H_{IntType}$	The energy consumed or released by an Interaction of this type. Negative for an exothermic Interaction Type in which energy is released.
$FwdActivnEnergy$ or E_{aFwd}	Energy required to enable an Interaction to proceed in the forward direction
$RevActivnEnergy$ or E_{aRev}	Energy required to enable an Interaction to proceed in the reverse direction
$FwdRateConst.$ or k_f	Forward Rate Contant, calculated as follows: Construction: $k_f = A_2 e^{\frac{-E_{aFwd}}{RT}}$ Fission: $k_f = A_1 e^{\frac{-E_{aFwd}}{RT}}$
$Rev.RateConst.$ or k_r	Reverse Rate Constant, calculated as follows: Construction: $k_r = A_1 e^{\frac{-E_{aRev}}{RT}}$ Fission: $k_r = A_2 e^{\frac{-E_{aRev}}{RT}}$
$Reactant_1$ or R_1	First Reactant Molecule Type
$Reactant_2$ or R_2	Second Reactant Molecule Type
$Product_1$ or P_1	First Product Molecule Type
$Product_2$ or P_2	Second Product Molecule Type

Table 3.6: Interaction Type Characteristics

Activation Energies	
Construction	$E_{aFwd} = E_{aMin2}$
	$E_{aRev} = E_{aMin1} + H_{IntType}$
Fission	$E_{aFwd} = E_{aMin1}$
	$E_{aRev} = E_{aMin2} + H_{IntType}$

Table 3.7: Activation Energies

General Parameters For Interaction Types	
A_1	Arrenhius Frequency Factor for Unimolecular Interaction Types
A_2	Arrenhius Frequency Factor for Bimolecular Interaction Types
E_{aMin1}	Minimum Activation Energy for Unimolecular Interaction Types
E_{aMin2}	Minimum Activation Energy for Bimolecular Interaction Types
R	Gas Constant

Table 3.8: General Parameters For Interaction Types

3.3.2 Interactions

An Interaction is a single instance of an Interaction Type taking place in the Reactor. The characteristics of an Actual Interaction are the same as those of the Interaction Type. Each Interaction that takes place results in the removal of one Molecule of each of the Interaction Type's Reactants from the Reactor, and the addition of one Molecule of each of the Products.

3.3.3 Actualised Molecule Types

A Molecule Type for which at least one Molecule currently exists in the Reactor. A Molecule Type that is not Actualised becomes Actualised whenever a Molecule of the type is added to the Reactor. A Molecule Type can become Actualised many times during a simulation run.[2]

The characteristics of an Actualised Molecule Type are as for the corresponding Realised Molecule Type.

3.3.4 Realised Molecule Types

A Molecule Type for which at least one Molecule has existed in the Reactor during a run is a Realised Molecule Type. It remains a Realised Molecule Type for the rest of the run, even if all of its Molecules are removed.

The characteristics of a Realised Molecule Type are shown in Table 3.9.

3.3.5 Realisable Interaction Types

A Realisable Interaction Type is an Interaction Type that currently is feasible in the Reactor. Here, 'feasible' usually means that at least one Molecule of each Reactant Molecule Type is present and available in the Reactor. In the less usual case of a Construction with both Reactant Molecule Types the same, two Molecules of this type must be present. A particular Interaction Type may be Realisable at some times during a model run, and not realisable at other times.

The characteristics of a Realisable Interaction Type are shown in Table 3.10.

[2] In situations where all Molecules of the type disappear from the Reactor, and another is subsequently added.

Realised Molecule Type Characteristics	
Name	**Description**
Identifier	Same as the identitier for the Molecule Type
Name	Optional name
Concentration	Number of Molecules of this type per unit volume in the Reactor

Table 3.9: Characteristics of a Realised Molecule Type

Realised Interaction Type Characteristics	
Name	**Description**
Identifier	Same as the *Identifier* of the Interaction Type
Category	Same as for the corresponding Interaction Type
Rate	The number of Actual Interactions of this type taking place in the Reactor during each timestep. *Rate* is calculated as follows: Construction $Rate = max((R_{Fwd} - R_{Rev})$ $R_{Fwd} = k_f[\mathrm{R}_1][\mathrm{R}_2]V$ $R_{Rev} = k_r[\mathrm{P}]V$ Fission $Rate = max((R_{Fwd} - R_{Rev}), 0)$ $R_{Fwd} = k_f[\mathrm{R}]V$ $R_{Rev} = k_r[\mathrm{P}_1][\mathrm{P}_2]V$ Note: The use of 'max' above is a simulation performance optimisation. It avoids modelling forward and reverse interactions that cancel out.

Table 3.10: Characteristics of a Realisable Interaction Type

The notation [X] denotes the concentration of Molecule Type X. The Reactor Volume (V) appears in the formulae for R_{Fwd} and R_{Rev} because terms such as $k_f[\mathrm{R}_1][\mathrm{R}_2]$ in the usual rate formulae determine the rate of change of *concentration*, whereas *Rate* here is

the the rate of Actual Interactions in the Reactor.

SimSoup calculates Interaction Rates as the difference between the forward rate and the reverse rate. Negative forward rates are not allowed; the reverse of a Fission C $\rightarrow$ A + B is considered to be a separate Interaction Type, the Construction A + B $\rightarrow$ C.

3.4 Outline Of Simulation Operation

This section outlines the overall operation of the SimSoup simulation. Section 3.4.1 describes the way the simulation scenario is setup using Action Requests. Section 3.4.2 summarises the operation of the main simulation loop. Section 3.4.3 provides further detail on the open ended network discovery process mentioned in Section 1.2.

3.4.1 Scenario Specification: Action Requests

SimSoup uses *Action Requests* to a specify the scenario for a simulation run. An Action Request specifies some action to be requested for execution by the simulator at a particular time or at repeated time intervals. The available Action Requests are as in Table 3.11.

Action Requests	
Name	**Description**
Add AtomType	Add a specified Atom Type
Add BondType	Add a specified Bond Type
Add_Derived_MolType	Add a new Molecule Type based on an existing ('base') Molecule Type
Add MolType	Add a Molecule Type
Add Molecule	Add a Molecule of a specified Type
Add JoinedMolType	Add a Molecule Type formed by joining two specified Molecule Types
Set SimParams	Set simulation parameters, including Reactor Characteristics as listed in Table 3.1 and general parameters for Interaction Types as in Table 3.8

Table 3.11: Action Requests

Execution of an Action Request does not necessarily succeed. For example, if an Add Molecule Action Request is scheduled to occur before the requested Molecule Type has been created, the Add Molecule Action Request is not executed and a warning is issued.

Similarly, an Add Molecule Request can only be executed if the required Atom Types and Bond Types have been defined by executing the appropriate Action Requests.

Material can only enter the Reactor as a result of the execution of an Add Molecule Action Request. Once Molecule(s) have been added, Fissions and Constructions can take place, resulting in new Molecules of new types.

In short, before anything of interest can happen in the Reactor, Action Requests must be setup and executed such that Atom Types and Bond Types are defined, along with at least one 'starter' Molecule Type. In addition, Action Request(s) must have been executed to add Molecule(s) to the Reactor.

3.4.2 Simulation Processing

SimSoup is a time-stepping simulation. At each timestep, the following procedure is undertaken:

- Check whether any Action Requests are due, and if so attempt to execute them. In the case of an Add_Molecule Action Request, Discover Interaction Types and Molecule Types as described in Section 3.4.3
- For each Realisable Interaction Type:
  - Calculate the number of Interactions in this timestep, calculating *Rate* as in Table 3.10
  - Remove the Reactants and add Product Molecules. Discover Interaction Types and Molecule Types as described in Section 3.4.3
- Remove 'leakage' Molecules
- Increment the Reactor Time

3.4.3 Open Ended Network Discovery Process

At the start of a simulation run, there are no Molecule Types or Interaction Types known in the Chemistry, and no Molecules in the Reactor. Typically, Action Requests to setup some 'starter' Molecule Types will have been defined in the simulation input, along with Action Requests to add Molecules of these types to the Reactor. Until at least one Molecule is added to the Reactor, no Interactions can take place.

When a Molecule Type is Actualised, new Interaction Types may become possible. SimSoup checks for Constructions in which Molecules of the Actualised type can join with other Molecules present in the Reactor. SimSoup also identifies a Fission Interaction Type in which the new Molecule Type will split.

The algorithm for identifying Constructions, 'Construction Discovery', is described in Section 3.4.3.1. 'Fission Discovery', is described in Section 3.4.3.2.

3.4.3.1 Construction Discovery: Joining Molecule Types

Whenever a Molecule Type is Actualised, Constructions involving the Actualised Molecule Type may become possible. SimSoup checks other Actualised Molecule Types to determine whether they can join with the newly Actualised Molecule Type. If a second Molecule of the newly Actualised type subsequently appears, it is checked if necessary to determine whether it can join with itself.

A Molecule Type can be Actualised multiple times. SimSoup ensures that any particular Reactant pair is only checked once during the course of a Simulation.

The key steps in determining whether two Reactants can join are as follows:

- Check that each Reactant has at least one Bonding Site; that is, an Atom whose electron shells are not full
- If a Maximum Molecule Mass ($MolMass_{Max}$)[3] has been specified, check whether the Product of the Join would exceed this mass. If it would not...
- Identify the eight relative orientations of the two candidate Reactants. These are identified by rotating one of the Reactants through 0°, 90°, 180° and 270° to give the first four relative orientations, and then reflecting in a horizontal axis and repeating the rotation process to get the other four relative orientations
- For each of the relative orientations
  - Scan one Reactant 'over' the other (without any further rotation or reflection) to check for any positionings in which the following conditions are both true:
    - At least one Bonding Site in the first Reactant is adjacent to at least one Bonding Site in the second Reactant
    - No two Atoms occupy the same space; each Board location contains a maximum of one Atom
    - A Bond Type has been defined for the two Atom Types.
  - The above procedure can potentially identify several feasible join configurations. The enthalpy change, $\Delta H_{IntType}$, for each such configuration is the total energy of the bonds that would be formed by the join. The configuration with the highest $\Delta H_{IntType}$, if any, is identified (or if there are several with the same highest $\Delta H_{IntType}$, the first to be found is identified)
  - If a join configuration was found, and its $\Delta H_{IntType}$ is higher than that for any previously identified join configuration for any relative orientation, then record this configuration/orientation and its $\Delta H_{IntType}$
- At this stage, all of the join configurations for all of the relative orientations have been checked. If no feasible join configuration was found then no Construction is possible. Otherwise, it is concluded that the Reactants can join in a Construction with the relative orientation, join configuration and $\Delta H_{IntType}$ previously recorded.

3.4.3.2 Fission Discovery: Splitting Molecule Types

Whenever a Molecule Type is Actualised, SimSoup checks whether it has been previously Actualised. That is, whether it is already a Realised Molecule Type. If it is not, then SimSoup determines how it will split.

As a model simplification, it is assumed that a Molecule Type will only split in one way. The Products are identified by finding a set of Bonds that, if broken, would split the Molecule into two unconnected parts. Where there are multiple ways in which the Molecule Type can be split, as is usually the case, the split identified is one that entails breaking a set of bonds with the lowest possible total energy. If there are several such 'least energy cuts', then one is chosen arbitrarily.

The splitting algorithm works on Molecule Types of arbitrary complexity.[4] It uses techniques of graph theory, [5] and in particular the Dijkstra shortest paths algorithm.

4 SimSoup User Manual

This section presents the User Manual for SimSoup.

The scenario for a simulator run is defined in an input file which must be loaded before the run starts. Section 4.2 specifies the content and format of the input file.

To start, stop and monitor a simulation run, the user interacts with SimSoup through a Graphical User Interface. Section 4.3 describes this user interface.

Links for download and installation are provided in Section 4.1.

[3] $MolMass_{Max}$ can be used to make Construction Discovery more computationally tractable. This can be useful in situations where there are many alternative ways in which Molecule Types can join. Ideally, given unlimited computation power, $MolMass_{Max}$ would not be used.

[4] Except for Molecule Types with a single Atom, which cannot be split by a chemical process!

[5] The SimSoup implementation makes use of the Boost Graph Library code.

Note: If you simply want to get started with SimSoup without reading all the documentation first, then just download, install and run SimSoup, and then follow the 'Quick Start' instructions on the Help Tab.

4.1 Download and Installation

SimSoup is a Linux program written in C++. It is provided in source code form, and can be downloaded from http://www.simsoup.info/SimSoup.htm

Installation instructions are provided. SimSoup has been developed on the Debian Squeeze distribution, but will on most modern Linux distributions.

4.2 SimSoup Scenario Input

As described in section 3.4.1, the scenario for a simulation run is specified using Action Requests. This section details how the Action Requests are specified in the SimSoup input.

Section 4.2.1 describes overall conventions for the input file. Section 4.2.2 and its subsections describe each Action Request and identify its Attributes. An Attribute can be used by more than one Action Request. The Attributes and their Data Types are therefore described separately, in Section 4.2.3.

4.2.1 Overall Input File Conventions

The input file includes *Input Records* and *Comments*. The following subsection describes Input Records. Comments are described in section 4.2.1.2.

4.2.1.1 Input Records

Each Input Record specifies one Action Request. It begins with the *Action Request Type*, and this is followed by one or more *Attribute Specifications*. Each Attribute Specification consists of an *Attribute Type* followed by an *Attribute Value*.

An Input Record is terminated by a semicolon, and may span multiple lines. Action Request Types and Attribute Types must be contiguous text strings (ie with no separating spaces or other 'whitespace'). Attribute Types must be prefixed with an '@' character. Attribute Values follow syntax rules specific to each Attribute Type. In some cases, (@Structure, @PertConfs, @MolTp_OpList and @Colour) Attribute Values consist of multiple items.

Attributes can be mandatory or optional, and can occur in any order within an Action Request Specification. Every Action Request must have an @Time Attribute.

An Attribute has a Data Type. For example, a @Time Attribute must have a non-negative integer value. Data Types for multi-item Attributes are more complex, and special syntax rules apply.

4.2.1.2 Comments

Comments may be included anywhere in the input file. A comment begins with '//', and continues to the end of the current line. Comments are otherwise ignored by the input processor. They provide a means for documenting the Action Request input.

4.2.2 Action Requests

This section presents each of the Action Requests, along with their Attributes. The format / syntax rules for the Attributes mentioned are specified in section 4.2.3.

4.2.2.1 Add_AtomType

Add_AtomType requests the addition of a specified Atom Type to the Chemistry.

Add_AtomType Action Request		
Attribute	**Description**	**Mandatory**
@Time	Time at which the Action Request should be executed	Yes
@Name	*Name* of the Atom Type	Yes
@Symbol	*Symbol* for the Atom Type	Yes
@Mass	M_{AType} for the Atom Type	Yes
@NumValElec	$N_{ValElec}$ for the Atom Type	Yes
@CapShell	C_{Shell} for the Atom Type	Yes
@Colour	*Colour* of the Atom Type	Yes

Table 4.1: Add_AtomType Action Request

Table 4.1 shows the Attributes for Add_AtomType. See Section 3.2.1 for further details of the meanings of the Atom Type Attributes mentioned.

The following shows example input for an Add_AtomType Action Request:

```
Add_AtomType
    @Time 1
    @Name Hydrogen
    @Symbol H
    @Mass 1.008
    @NumValElec 1     // One valence electron
    @CapShell 2       // Shell capacity 2
    @Colour 100:100:100; // Display white
```

4.2.2.2 Add_BondType

Add_BondType requests the addition of a specified Bond Type to the Chemistry.

Add_BondType Action Request		
Attribute	**Description**	**Mandatory**
@Time	Simulation Time at which Action Request is to be executed	Yes
@Atom1	$AType_1$	Yes
@Atom2	$AType_2$	Yes
@Order	$BondOrder$	Yes
@Enthalpy	$H_{BondDefault}$	Yes
@PertConfs	$PertConfs$	No

Table 4.2: Add_BondType Action Request

Table 4.2 shows the Attributes for Add_BondType. See Section 3.2.2 for the meanings of the Bond Type Attributes mentioned.

The following shows example input for an Add_BondType Action Request that does not include an @PertConfs Attribute:

```
Add_BondType
    @Time 2
    @Atom1 C
    @Atom2 H
    @Order 1
    @Enthalpy 412;
```

The following shows example input for an Add_BondType Action Request that includes an @PertConfs Attribute:

```
Add_BondType
    @Time 0
    @Atom1 p
    @Atom2 s
    @Order 1
    @Enthalpy 100
    @PertConfs -95, m:2:0 / -50, a:-1:0 # ;
```

The second Action Request specifies a single order Bond between Atom Type 'p' and Atom Type 's'. The Default Bond Enthalpy is 100. The @PertConfs Attribute indicates that there are two configurations in which a p-s Bond can be perturbed. In the first case, the Bond Enthalpy is reduced by 95 due to an 'm' Atom with $X_{Offset} = 2$ and $Y_{Offset} = 0$. In the second case the reduction is 50 due to an 'a' Atom. See Section 4.2.3.3 for details of the PertConfs Data Type.

4.2.2.3 Add_Derived_MolType

Add_Derived_MolType requests the addition of a new Molecule Type to the Chemistry, based on (derived from) a previously defined Molecule Type.

Add_Derived_MolType Action Request		
Attribute	**Description**	**Mandatory**
@Time	Simulation Time at which Action Request is to be executed	Yes
@Name	*Name* of the Derived Molecule Type	Yes
@Base_MolType	*Name* of the Molecule Type from which the new Molecule Type is derived	Yes
@MolType_OpList	List of *Operations* to be applied to the @Base_MolType to produce the new Molecule Type (@Name)	Yes

Table 4.3: Add_Derived_MolType Action Request

Table 4.3 shows the Attributes for the Add_Derived_MolType Action Request. The following shows an example input:

```
Add_Derived_MolType
    @Time 0
    @Name S1
    @Base_MolType S0
    @MolType_OpList
        SplitVert:31:1          /
        Set_BoardPos:2:32:aR    /
        Set_BoardPos:7:32:aR #;
```

The example indicates that at Time 0, Molecule Type S1 is to be created based on Molecule Type S0. Molecule Type S0 is first split; Atoms to the right of Column 31 are shifted one position to the right. The Atom Location at Row 2 Column 32 is set to contain an 'a' Atom with one Right Bond, and the Atom Location at Row 7 Column 32 is also set similarly. See Section 4.2.3.2 regarding Right Bonds and Down Bonds, and Section 4.2.3.1 for details of MolType_OpList.

4.2.2.4 Add_Molecule

Add_Molecule requests the addition of a specified number of new Molecules of a specified type to the Reactor. If required, the action can be repeated at specified intervals. This can be useful for setting up a constant supply of 'food' Molecules.

Table 4.4 shows the Attributes for Add_Molecule. Example input is shown below:

Add_Molecule Action Request		
Attribute	**Description**	**Mandatory**
@Time	Simulation Time at which Action Request is to be executed	Yes
@Num_To_Add	Number of Molecules to add	Yes
@Name	Molecule Type Name	Yes
@Repeat_Time	Simulation Time between repeats	No

Table 4.4: Add_Molecule Action Request

```
Add_Molecule
    @Time 1000
    @Num_To_Add 200
    @Name Methane
    @Repeat_Time 100;
```

4.2.2.5 Add_MolType

Add_MolType requests the addition of a new Molecule Type to the Chemistry.

Add_MolType Action Request		
Attribute	**Description**	**Mandatory**
@Time	Simulation Time at which Action Request is to be executed	Yes
@Name	Molecule Type Name	Yes
@Structure	Molecular structure. See Sections 4.2.3.2 and 3.2.3	Yes

Table 4.5: Add_MolType Action Request

Table 4.5 shows the Attributes for the Add_MolType Action Request. Example input is shown below:

```
Add_MolType
    @Time 0
    @Name Methane
    @Structure 1:HD     /
               HR:CR:H  /
               1:H # ;
```

The @Name Attribute is mandatory, and enables a subsequent Add_Molecule Action Request to add Molecules of this type. An Add_Molecule_Type Action Request can also be used in scenarios where Molecules of the type will be generated spontaneously as the simulation runs, rather than being externally input. The Molecule Type name will then be visible on the SimSoup user interface.

4.2.2.6 Add_Joined_MolType

Add_Joined_MolType requests the addition of a new Molecule Type formed by joining two previously defined Molecule Types.

Add_Joined_MolType Action Request		
Attribute	**Description**	**Mandatory**
@Time	Simulation Time at which Action Request is to be executed	Yes
@MolType1	Name of the first Molecule Type to join	Yes
@MolType2	Name of the second Molecule Type to join	Yes
@Joined_MolType	Name of the joined Molecule Type	Yes

Table 4.6: Add_Joined_MolType Action Request

The Attributes for Add_JoinedMolType are shown in Table 4.6. Example input is shown below:

```
Add_Joined_MolType
    @Time 0
    @MolType1  R0
    @MolType2  R0p
    @Joined_MolType R0.R0p;
```

4.2.2.7 Set_SimParams

Set_SimParams requests the various simulation parameters to be set to specified values. Table 4.7 shows the Attributes. In SimSoup version 0.6, some of the parameters only take full effect if specified before any Interactions take place. It is recommended that Set_SimParams should be the first Action Request specified, and that it should have @Time = 0.

Example input for a Set_SimParams Action Request is shown below:

```
Set_SimParams
    @Time 0
    @Reactor_Temp  250.0
    @Reactor_Leakage 0.001
    @Reactor_Vol 5000
    @Stats_Coll_Intvl 50
    @Display_Upd_Interval 2
```

Set_SimParams Action Request		
Attribute	**Charactersistics**	**Mand.**
@Time	Simulation Time at which Action Request is to be executed	Yes
@Reactor_Temp	Reactor Temperature, T	Yes
@Reactor_Leakage	*ReactorLeakage*	Yes
@Reactor_Vol	Reactor Volume, V	Yes
@Stats_Coll_Intvl	Statistics Collection Interval (simulation timesteps)	Yes
@Display_Upd_Interval	Display Update Interval (seconds)	Yes
@Arren_Freq_Factor_UniMol	A_1	Yes
@Arren_Freq_Factor_BiMol	A_2	Yes
@Default_Min_ Activation_Energy_BiMol	E_{aMin1}	Yes
@Default_Min_ Activation_Energy_UniMol	E_{aMin1}	Yes
@Gas_Constant	R	Yes
@Max_MolMass	$MolMass_{Max}$	Yes

Table 4.7: Set_SimParams Action Request

```
@Arren_Freq_Factor_UniMol 1.0
@Arren_Freq_Factor_BiMol  0.1
@Default_Min_Activation_Energy_BiMol  0.0
@Default_Min_Activation_Energy_UniMol 0.0
@Max_MolMass 0.0     // Do not limit Mass
@Gas_Constant  0.008314;
```

A_1, A_2, E_{aMin1}, E_{aMin2} and R are described in section 3.3.2. See section 3.1 regarding Reactor Temperature, Reactor Volume and Reactor Leakage, and Section 3.4.3.1 regarding $MolMass_{Max}$. Section 4.3.4.3 describes Statistics Collection Interval and Display Update Interval.

4.2.3 Attributes and their Data Types

The Action Requests described in Section 4.2.2 each specify various Attributes.

Table 4.8 lists the different Attributes, and indicates the Data Type used in each case. Each Data Type is used for a particular kind of data, and the input must conform to corresponding syntactic rules.

Attribute Data Types	
Attribute	**Data Type**
@Arren_Freq_Factor_BiMol	PosFloat
@Arren_Freq_Factor_UniMol	PosFloat
@Atom1	String1
@Atom2	String1
@Base_MolType	String
@BondType	BondType
@CapShell	PosInt
@Colour	RGBString
@Default_Min_Activation_Energy_BiMol	NonNegFloat
@Default_Min_Activation_Energy_UniMol	NonNegFloat
@Display_Upd_Interval	PosInt
@Enthalpy	PosFloat
@Gas_Constant	PosFloat
@Joined_MolType	String
@Mass	PosFloat
@Max_MolMass	NonNegFloat
@MolType1	String
@MolType2	String
@MolType_OpList	MolTp_OpList
@Name	String
@Num_To_Add	PosInt
@NumValElec	PosInt
@Order	PosInt
@PertConfs	PertConfs
@Reactor_Leakage	PosFloat
@Reactor_Temp	PosFloat
@Repeat_Time	PosInt
@Reactor_Vol	PosFloat
@Stats_Coll_Intvl	PosInt
@Structure	MolType_Struct
@Symbol	String1
@Time	NonNegInt

Table 4.8: Attribute and their Data Types

Table 4.9 describes the Data Types themselves, and indicates the syntax rules in the case of simple Data Types. The following subsections describe the syntax rules for the more complex Data Types.

4.2.3.1 MolTp_OpList Data Type Input

A MolTp_OpList is a list of operations to be applied to a Molecule Type. It is used with the Add_Derived_MolType Action Request to specify how a new Molecule Type will be derived from an existing ('base') Molecule Type. In SimSoup 0.6, there are two Operation Types: `SplitVert` and `Set_BoardPos`.

Data Types for SimSoup Input	
Data Type	**Description**
MolTp_OpList	A Molecule Type Operation List. See section 4.2.3.1
MolType_Struct	A specification of a Molecule Type structure. See section 4.2.3.2
NonNegFloat	A non-negative floating point number
NonNegInt	A non-negative integer
PertConfs	A specification of a set of Perturbation Configurations for a Bond Type. See section 4.2.3.3
PosFloat	A positive floating point number
PosInt	A positive integer
RGBString	A specifiction of a colour in RGB ('Red-Green-Blue') format, 'rrr:ggg:bbb', where 'rrr', 'ggg' and 'bbb' are each an integer from 0 to 100
String	A contiguous string of text
String1	A single character

Table 4.9: Data Types for SimSoup Input

SplitVert
The `SplitVert` operation splits a Molecule Type along a vertical axis, separating the two parts by a specified number of 'board' spaces. The result is not in itself a viable Molecule Type. A Molecule Type must always form a single component with no separate or 'unconnected' parts. The `Set_BoardPos` operation must be used with `SplitVert` to 'fill in' the gaps.

A `SplitVert` operation specifier has three parts:

- The `SplitVert` keyword
- The Split Column
- The Split Shift Size.

The `SplitVert` operation shifts all Atoms in the Molecule Type after the Split Column to the right by the Split Shift Size.

Set_BoardPos
The `Set_BoardPos` operation makes an amendment to to a Molecule Type at a specified location in the structure. A Set_BoardPos operation specifier has the following parts:

- The `Set_BoardPos` keyword
- The Atom Location Row
- The Atom Location Column
- An Atom Location Specifier (see Section 4.2.3.2), or '1' to indicate an unoccupied Atom location.

MolTp_OpList Format
Operations in a MolTp_OpList are separated with an '/' character. The end of the MolTp_OpList is indicated by a '`#`' character. Items within a particular operation specifier must be separated with colons. See Section 4.2.2.3 for an example.

4.2.3.2 MolType_Struct Data Type Input
A MolType_Struct is a specification of the structure of a Molecule Type. A MolType_Struct consists of a set of Rows, each separated from the previous Row by a '`/`' character. The end of a MolType_Struct is delimited by a '`#`' character.

A Row consists of tokens, each of which is an *Atom Location Specifier*, or an *Atom Spacing Specifier*. An Atom Location Specifier includes the following:

- An Atom Type Symbol
- An optional *Right Bond Specifier*
- An optional *Down Bond Specifier*.

The Atom Type Symbol must be the value of @Symbol from an Add_AtomType Action Request.

A Right Bond Specifier begins with an 'R' character. This may optionally be followed by the number '2' or '3'. The 'R' signifies that the Atom is bonded to the Atom to its right. If there is no number following the 'R', the Bond has Order 1. If there is a number, it specifies the Bond Order.

A Down Bond Specifier is similar to a Right Bond Specifier, except that it begins with a 'D' character. The following are valid Atom Location Specifiers:

- `H`
- `aR`
- `HR2`
- `CRD`
- `CRD3`
- `aR2D3`.

Note that there are no 'Up Bond' or 'Left Bond' specifiers. It is always the Atom on the left of or above its Bond partner that has the Bond Specifier.

An Atom Spacing specifier is a positive integer that specifies the horizontal spacing between two Atoms in the structure, or, in the case of the first Atom in a

Row, a number of empty spaces to its left. Multiple consecutive Atom Spacing Specifiers are equivalent to a single Atom Spacing Specifier for the sum of the spaces. For example, `'1:1:1:1:'` is equivalent to `'4:'`. The first format can be useful for complex molecular structures, which can be edited and output in CSV format using a spreadsheet program.

Figure 4.1 shows the Molecule Type specified by the MolType_Struct `1:ORD:CR3:CR:C/HR:NR:O #`.

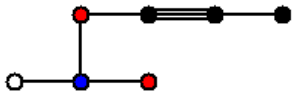

Figure 4.1: The Molecule Type specified by `1:ORD:CR3:CR:C/HR:NR:O #`. The red Atoms have Symbol 'O', the blue Atom has Symbol 'N', the white Atom has Symbol 'H', and the black Atoms have Symbol 'C'.

4.2.3.3 PertConfs Data Type Input

A PertConfs Attribute is a specification of a set of Atomic configurations that weaken a Bond. See Section 4.2.2.2 for an explanation.

The SimSoup input for a PertConfs Attribute includes the following items:

- ΔH_{Bond}: A floating point number
- One or more groups of...
  - *AType*: An Atom Type Symbol
  - X_{Offset}: An integer
  - Y_{Offset}: An integer

The items within each *AType* / Offset grouping are separated with colons. The individual groupings are separated from each other and from ΔH_{Bond} by commas.

The @PertConfs Attribute is terminated with a # character. The semicolon following the # character in Section 4.2.2.2 is the usual terminator for an Action Request.

4.3 The Graphical User Interface

SimSoup has a graphical user interface that can be used to:

- Initiate processing of the input file
- Start and stop the simulation
- View simulation progress
- View details of the state of the Chemistry and the Reactor
- View plots based on time series data collected as the simulation runs.

The user interface has four main sections (tabs) as follows:

- The Scenario Tab
- The Chemistry and Reactor Views Tab
- The Simulation Statistics Tab
- The Help Tab.

4.3.1 The SimSoup Menu

The SimSoup Menu system is very simple. It has the following structure:

- File
  - Open Scenario
  - Quit
- Simulation
  - Run Simulation
  - Stop Simulation.

Open Scenario loads and processes a scenario input file. Run Simulation and Stop Simulation start and stop simulation processing. Note that Stop Simulation allows the current simulation timestep to complete before processing stops.

4.3.2 The Scenario Tab

Figure 4.2 shows an example of the Scenario Tab. It has three sections:

- The Action Request Schedule List
- The Simulation Log
- The Simulation Processing Status View.

The Action Request Schedule List is on the left hand side of the Scenario Tab, and shows the Action Requests that have been successfully input.

The Simulation Log is on the top right of the Scenario Tab. If there are any errors on processing the scenario input file, these are shown. They must be corrected before the main simulation processing can be started.

Once simulation processing has started, the Simulation Log shows the Action Requests that have been executed. It also displays warnings for any Action Requests that could not be executed.

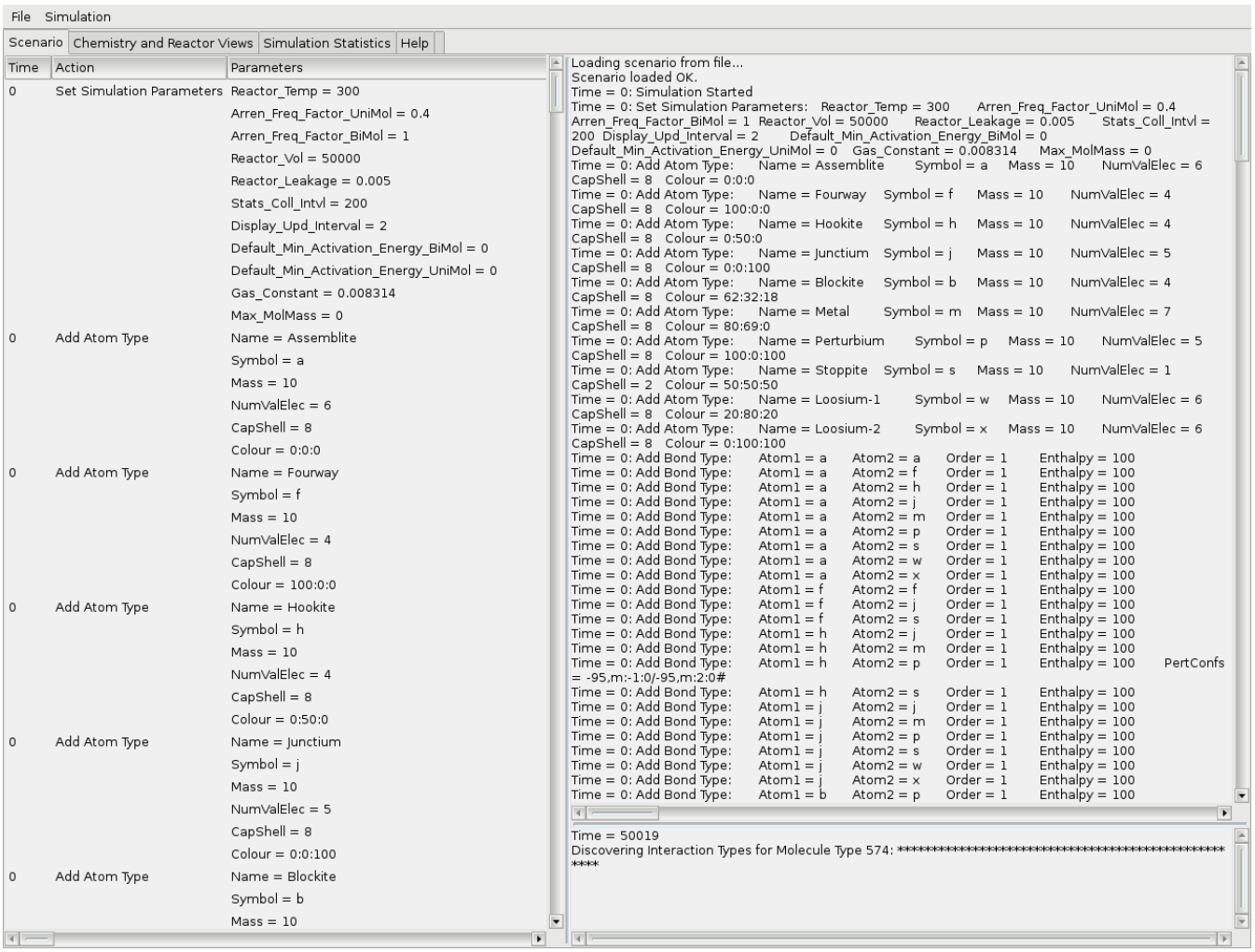

Figure 4.2: The Scenario Tab

The Simulation Processing Status View is at the bottom right of the Scenario Tab. During simulation processing, it show the Simulation Time, and indicates the nature of current processing. This provides useful feedback during periods of intensive processing when there may be no other visible output on the user interface.

4.3.3 The Chemistry and Reactor Views Tab

Figure 4.3 shows the Chemistry and Reactor Views Tab.

The various parts of the display can be resized by clicking on and dragging the separators between the areas. If the separators are not easily visible on your system, try changing your desktop settings; the example in Figure 4.3 was produced using the Xfce-winter GTK theme.

4.3.3.1 The Chemistry View

The Chemistry View is on the left of the Chemistry and Reactor Views Tab. It shows details of the Molecule Types and Interaction Types in the Chemistry. Figure 4.4 shows an example, with the Reactor View hidden by sliding the separator.

At the top are the Molecule Type List and the Interaction Type List. The Molecule Type List is on the left and shows all the Molecule Types that are known in the Chemistry. The Interaction Type List to the right shows Interaction Types involving the currently selected item from the Molecule Type List.

At the bottom are graphical displays. The one on the left shows the structure of the currently selected Molecule Type. The one on the right shows the Reactant(s) and Product(s) for the currently selected Interaction Type; the Reactant(s) are on the left, the Product(s) are on the right.

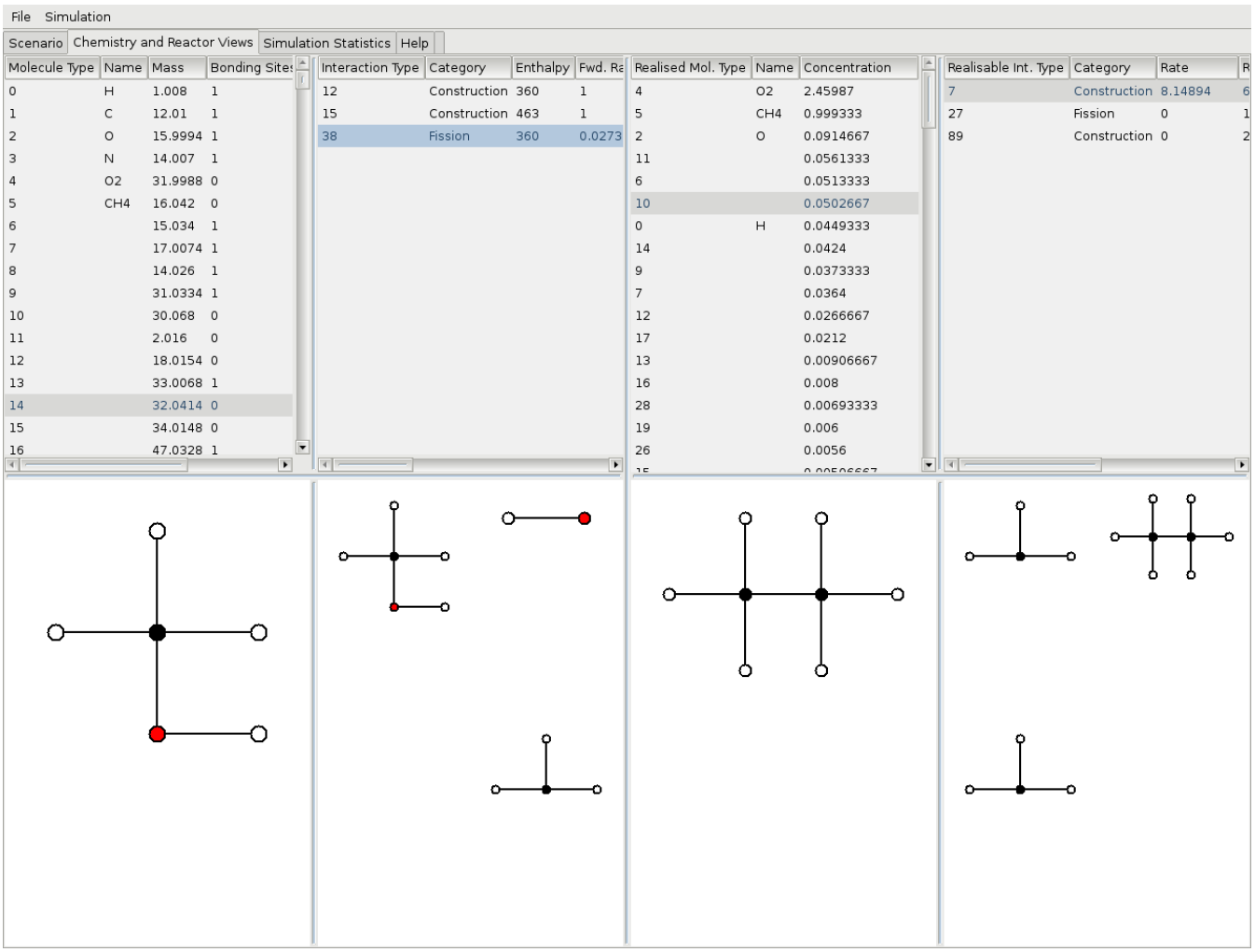

Figure 4.3: The Chemistry and Reactor Views Tab

4.3.3.2 The Reactor View

The Reactor View is on the right of the Chemistry and Reactor Views Tab. It shows details of the Realised Molecule Types (see Section 3.3.4) and Realisable Interaction Types (see Section 3.3.5) for the Reactor at the current Simulation Time. The layout is analogous to that of the Chemistry View. Figure 5.1 shows an example of the Reactor View.

4.3.4 The Simulation Statistics Tab

The Simulation Statistics Tab displays simulation statistics in two formats: Data Series Plots and the Manhattan Plot.

4.3.4.1 Data Series Plots

Figure 5.2 shows the Simulation Statistics Tab displaying a set of Data Series Plots. The main part of the display shows up to six plots of user selectable simulation variables.

To the left of the plot area (from top to bottom) are the following:

- The Display Group Area: This shows the two types of display available. In this case the Series Plots display has been selected. To display the Manhattan Plot, double click on Manhattan Plot
- The Series List Area: This shows the lists of data series that are available for display:
  - Interaction Counts: Rates of the various Interaction Types in the Reactor
  - Molecule Counts: Numbers of Molecules of the various types in the Reactor
  - Reactor Overview: Measures of overall Reactor activity

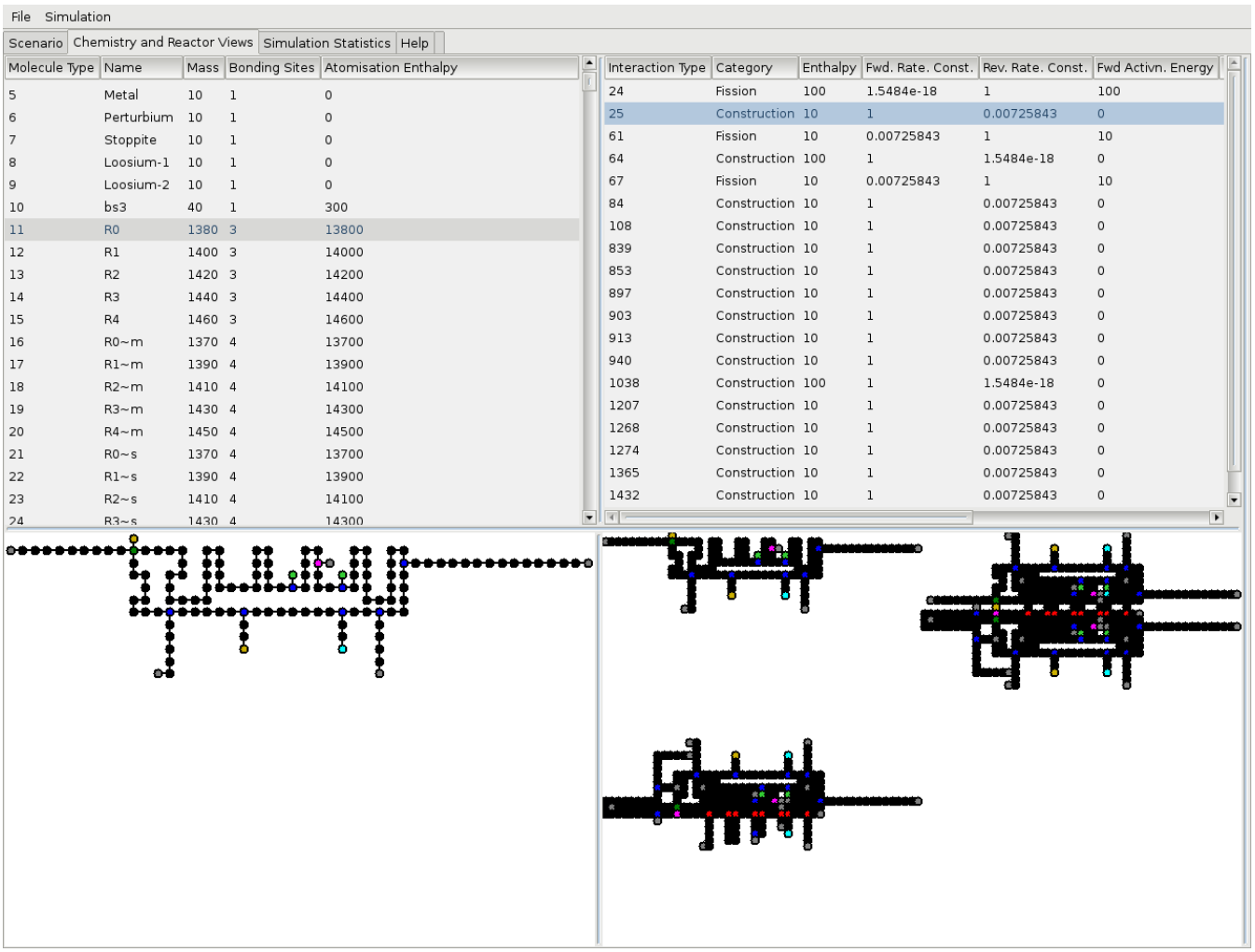

Figure 4.4: The Chemistry View

To display the various series within a particular series list, click on that list

- The Series Area: This displays the data series in the currently selected series list. To add or remove a series to / from the display, double click on that series. Up to six series can be displayed simultaneously.

4.3.4.2 The Manhattan Plot

Figure 5.3 shows the Simulation Statistics Tab displaying a Manhattan Plot.

The Manhattan Plot provides a graphical display showing how the overall content of the Reactor changes over time. The tone of each point indicates the difference between the Reactor Compositions (in terms of the numbers of Molecules of each type) at time t (x axis) and a time Δt (y axis) earlier. Black indicates zero difference. White indicates a substantial difference. The dark triangles indicate periods during which the Reactor Composition is roughly constant. The right hand edge of a triangle indicates a time at which the composition changes sharply.

4.3.4.3 Controlling Display Updates

The following Simulation Parameters control the way simulation statistics are collected and displayed:

- @Stats_Coll_Intvl: Determines the number of simulation timesteps between times at which SimSoup collects data for Data Series Plots and the Manhattan Plot. Note that a higher interval will result in use of less system memory as the simulation runs
- @Display_Upd_Intvl: Controls the rate at which SimSoup refreshes the user interface. A low value ensures that the screen is refreshed even during intensive 'Interaction Discovery' processing.

4.3.5 The Help Tab

The Help Tab has three sub-tabs:

- Quick Start: A short guide to running SimSoup for the first time
- About: Version information
- Licence: The licence for use of SimSoup source code.

5 Acknowledgements

Thanks to Inman Harvey for inviting me to attend Alergic group seminars at Sussex University. Thanks also to Tom Froese and Matthew Egbert for arranging a talk in 2010 on Attractor Based Evolution in which I presented ideas that developed into the metabolism based memory systems described in subsequent papers. Finally, thanks to many others at Sussex, at ECAL, and elsewhere for numerous interesting and inspiring discussions.

Bibliography

Atkins, P. (2001). *The Elements of Physical Chemistry*. Oxford University Press, third edition.

Atkins, P. and Paula, J. D. (2006). *AtkinsT Physical Chemistry*. Oxford University Press, eighth edition.

Gordon-Smith, C. (2005). SimSoup: An artificial chemistry model for investigation of the evolution of metabolic networks. ECAL workshop paper. Available at c http://www.simsoup.info/Publications.html .

Gordon-Smith, C. (2007). Evolution without smart molecules. ECAL workshop paper. Available at http://www.simsoup.info/Publications.html .

Gordon-Smith, C. (2009a). The origin of life: A network oriented view. In *Proceedings - Levels of Selection and Individuality in Evolution: Conceptual Issues and the Role of Artificial Life Models, September 2009*.

Gordon-Smith, C. (2009b). SimSoup: Artificial chemistry meets Pauling. In *Advances in Artificial Life: 10th European Conference, ECAL 2009, Proceedings*, Lecture Notes in Computer Science. Springer-Verlag.

Gordon-Smith, C. (2011). Non-template molecules designed for open-ended evolution. In *Advances in Artificial Life, ECAL 2011 Proceedings of the Eleventh European Conference on the Synthesis and Simulation of Living Systems*.

Gordon-Smith, C. (2013). Simsoup: Molecules designed for switchable autocatalytic memory. In *Advances in Artificial Life, ECAL 2013 Proceedings of the Twelfth European Conference on the Synthesis and Simulation of Living Systems*.

Pauling, L. (1960). *The Nature Of The Chemical Bond*. Cornell University Press.

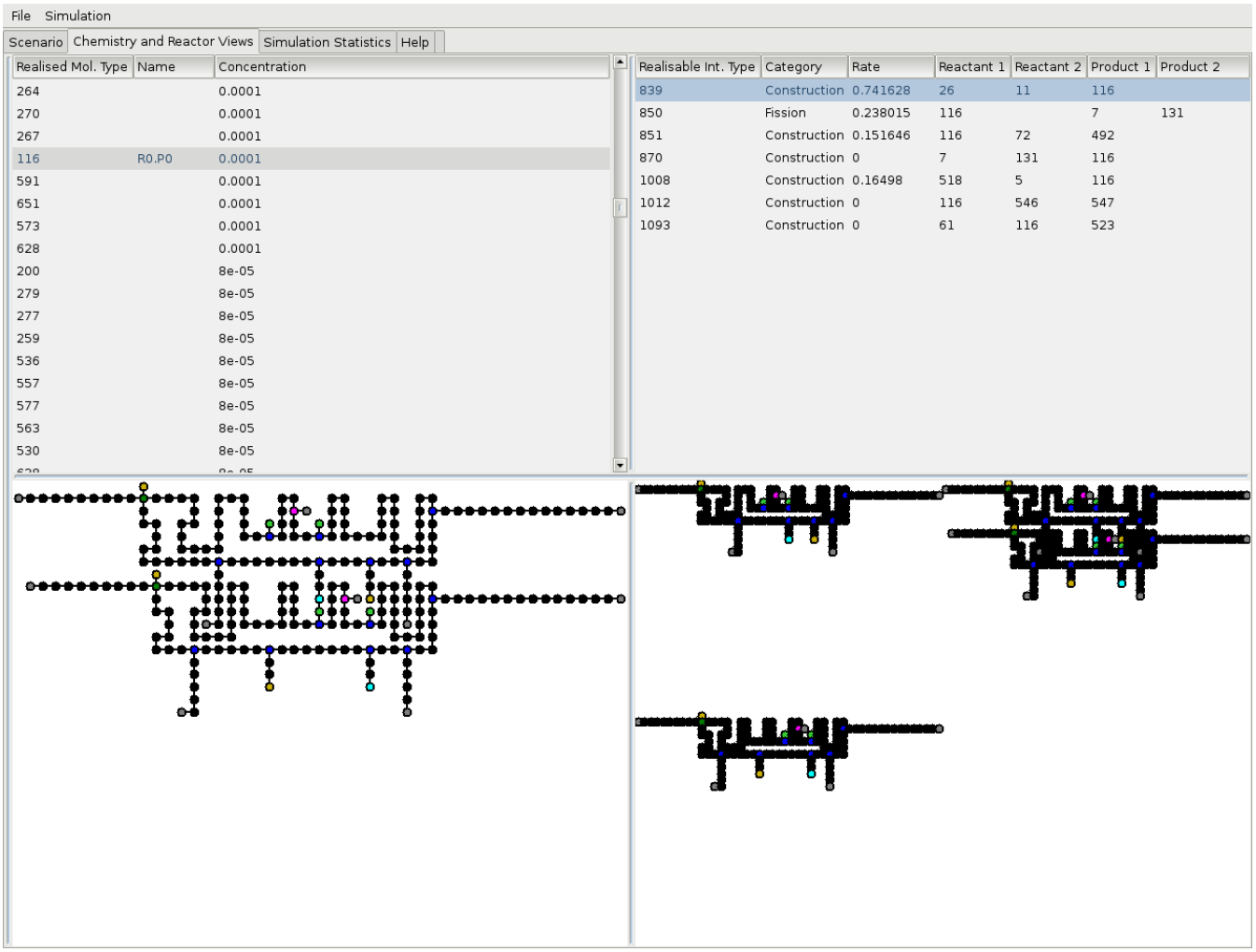

Figure 5.1: The Reactor View

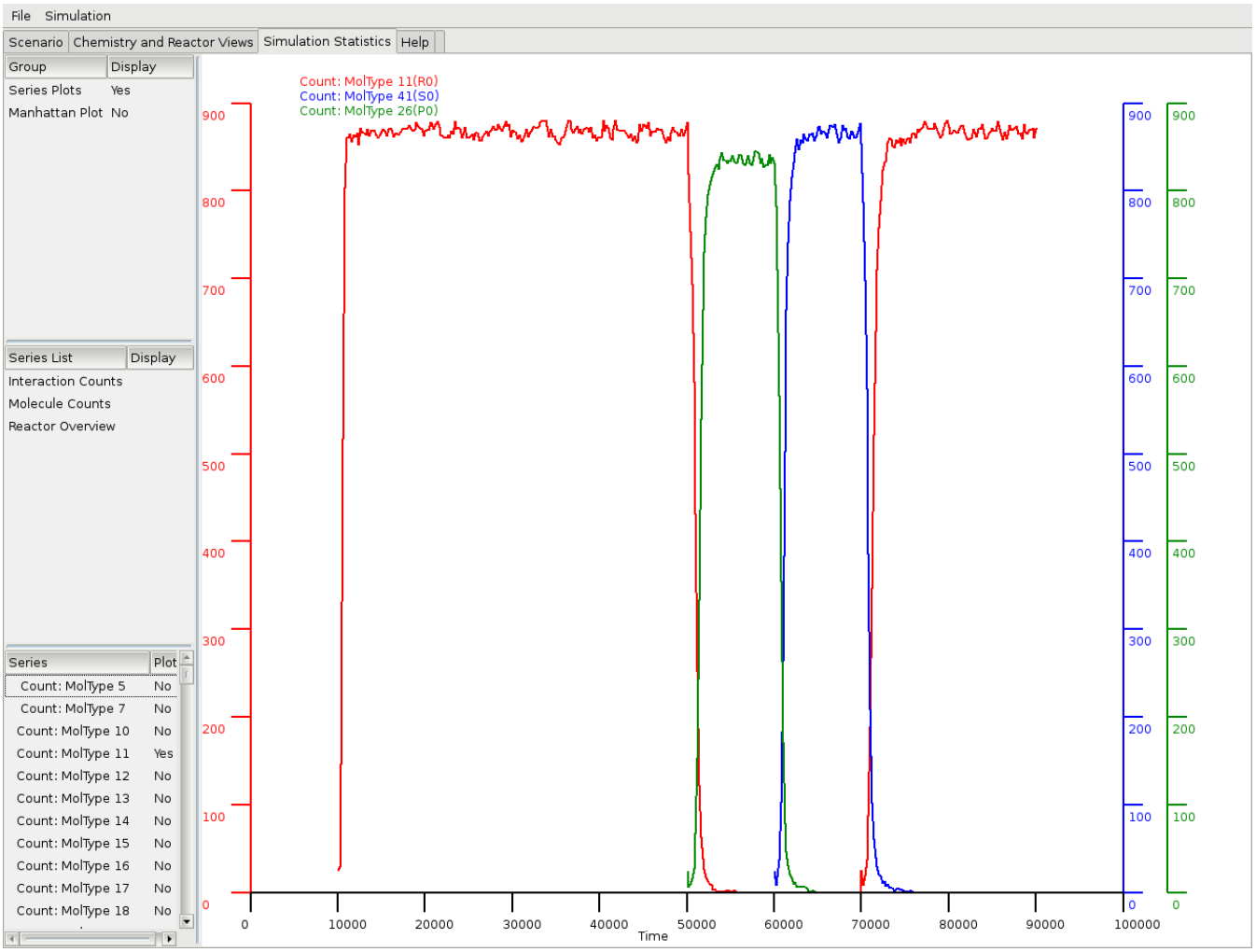

Figure 5.2: The Simulation Statistics Data Series Plot Screen

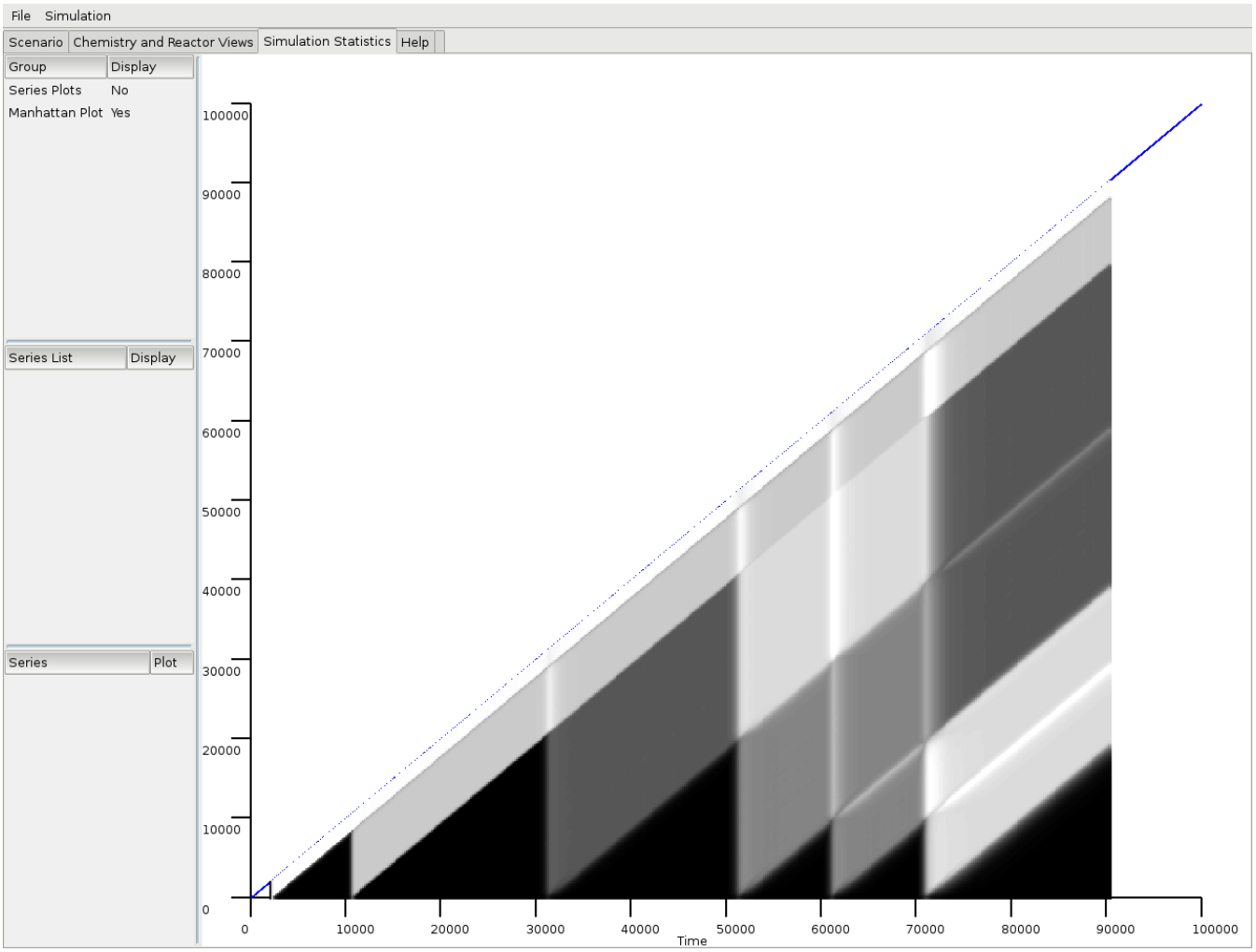

Figure 5.3: The Simulation Statistics Manhattan Plot Screen