Automatically Moving Between Levels in Artificial Chemistries

Adam Nellis and Susan Stepney

York Centre for Complex Systems Analysis, University of York, UK, Y010 5DD
adam@cs.york.ac.uk www.yccsa.org

Abstract

We introduce multi-level Artificial Chemistries as a way of tackling difficult problems in the evolution of complexity. We present two algorithms for moving between levels of abstraction in a multi-level Artificial Chemistry. (1) Moving upwards from a low-level description to a high-level description involves making approximations. We discuss these, and provide an algorithm to perform the approximations. (2) Moving downwards is more problematic. We discuss the issues involved in moving down, including conservation of mass. We present an algorithm to generate constraints that any low-level implementation of the system must satisfy. These constraints can be used to: obtain information about the system; automatically generate a low-level implementation of the system; guide a search for suitable low-level implementations of the system.

Introduction

Artificial Chemistries (AChems) can be explored from a computational viewpoint, for example, as tools for implementing evolutionary algorithms [9] and controlling robots [6]. They can also be used to model biological systems [10] such as replication [12] and membrane formation [13]. These varied applications of AChems lead to varied ways of defining them, and consequently to AChems defined on different levels of abstraction, with different properties. However, one common feature among AChems is that they are defined on only one level. Some problems, relevant to both computation and biology, span two or more levels of abstraction (for example, any of the 'major transitions in evolution' [14]). If AChems are to tackle these problems, they must span multiple levels of abstraction.

Previous authors have observed that biological systems contain components on different levels [3], but the purpose of multi-level AChems is to produce two different models of the same system, from two different levels. Work has been done on Course-Grained Molecular Dynamics [1] and Dissipative Particle Dynamics [11], which move from the very low level simulations of Molecular Dynamics, upwards to a slightly higher level that is more computationally tractable for larger molecules and longer timescales. But these systems still only operate on one level. Currently there is no well-defined way for the AChem itself to move between levels of abstraction. We discuss the issues involved in moving between levels of abstraction, and present two algorithms to aid movement up and down levels of abstraction in AChems.

Traditionally, people use computers to do the 'work' of running the AChem, and themselves do the 'meta-work' of deciding at which level to run. But what if computers could do this 'meta-work'? A system that could automatically decide which level to model at could attempt to tackle some of the difficult modelling challenges that span multiple levels, such as the 'major transitions in evolution'. Here we discuss both moving downwards from a higher level to a lower level and moving upwards from a lower level to a higher level.

The higher level is an *approximation* of the lower level. The lower level contains more information than the higher level, and so moving downwards requires adding this information into the system. When moving downwards, we do not know how the lower level is implemented. We only know how it must behave when viewed from a high level. So we cannot map directly from a high-level description to 'the correct' low-level description. In this paper, we map to a set of *constraints* that any low-level implementation must satisfy. These constraints describe how the low-level components of the system combine to form high-level structures.

The constraints could then be used to guide an implementation of the lower level. For some low-level implementations, these constraints correspond almost directly to an implementation (with possibly some arbitrary choices to be made). For more involved low-level descriptions, these constraints can be used to search for low-level implementations.

When moving up from a low-level description to a high-level description, an approximation must be made. The purpose of having a high-level description of a system is that there is too much information in the low-level description, and a summary of this information is desired. The high-level description approximates this information in a meaningful way. We must decide precisely how to approximate the system and how much to approximate it. An algorithm is presented for performing this approximation, and the issues surrounding approximation are discussed.

What is an Artificial Chemistry?

AChems are agent-based systems where the agents are analogues of chemicals participating in reactions. There are different types of AChem [4] with varying levels of complexity. The simplest are defined by finite lists of chemical types and the reactions they can participate in. More sophisticated AChems define chemicals containing some internal structure or properties. This makes it possible to describe an infinite number of different chemicals using a finite number of properties [10]. The reactions in these systems do not need to be explicitly listed; they are defined implicitly by the structure and properties of the chemicals, and specific reactions can be computed as and when they are needed.

When defining reactions implicitly, the possibility exists for open chemistries [7]. In an open chemistry, the possible chemical species that can exist need not be pre-specified. Although many different chemical species are possible, only a small number of them exist at any one time. A particular instance of the chemistry occupies a sub-space of the space of all possible chemical species. As an open chemistry runs, it changes the sub-space that it occupies.

If an AChem is to be used to evolve a network of chemicals and reactions, an open chemistry is required. Additionally, the chemistry should also be *evolvable*: the chemical species should change (via mutation) in a structured way that evolution can use to move through the space of possible chemical species. Most changes should have only a small effect (so a mutated chemical can perform the same reactions as its parent, but maybe faster or slower), but some changes should have a large effect (occasionally a mutated chemical can perform a new reaction, or lose the ability to perform an existing reaction).

One way of making evolvable chemistries is to use subsymbolic chemistries [5], where chemicals have two levels of description. On the higher level, the system is an open AChem with chemical species containing structure and rules that define their reactions. On the lower level, a chemical is composed of parts that interact to give rise to properties that entail the rules on the higher level. The lower level could be a complex system such as a random boolean network [5], it could be another AChem (for example a simple, closed chemistry), or it could be a computer programming language [8]. AChems that work on two or more levels have the potential to possess properties such as evolvability.

What are levels?

There is no 'correct' level at which to design AChems, as it depends on the particular problem being solved. This includes whether the purpose of using the AChem is to simulate a system from actual chemistry (or biochemistry), or to use the AChem as a computational tool, exploiting its properties to create a computational system (or to study a computational system). But there are some problems that involve crossing levels. For example, actual chemistry has gone through events crossing levels at different times during the evolution of life (the 'major transitions in evolution' [14]), for example: naked replicating molecules becoming encased in compartments and replicating as populations; RNA acting as both genes and enzymes, changing to use DNA as genes and proteins as enzymes; and the evolution of multi-cellular organisms from single-celled organisms. These kinds of problem may be interesting to systems biologists wanting to better understand what happened in real chemistry/biology. They may also be interesting to people wanting to use AChems for computational purposes, as they are examples of natural systems increasing their own complexity, something that current artificial systems find difficult to achieve.

All of these problems involve concepts at two (or more) levels. Choosing the most appropriate level at which to model is not easy. Addressing these problems (from the point of view of either biology or computation) involves one of two options: either modelling and simulating the whole system from the lower level, and enduring the computational burden that this entails; or modelling the system on both levels simultaneously, switching between the two levels in a multi-level chemistry. To automate the second option requires a well defined way of moving between the levels.

Going downwards

The concept of multi-level chemistries can provide new ways of thinking about high-level, symbolic, chemistries (lists of chemicals as symbols, and their reactions). Any symbolic chemistry describes a system at a certain level.

For systems that are models of the real world, there is always a lower level of description that the system could be described on (until we reach the level of our understanding of particle physics). Also, for real world systems, some information about this lower level is always known (we know that organisms are composed of cells, which are composed of molecules, and so on.)

For artificial systems, however, the implementation of any level is arbitrary (and is often chosen to make the program execute efficiently). So when describing an artificial system in terms of a lower level, there are arbitrary implementation choices to be made, some of which are constrained by the higher level. Looking for these constraints can give insight and information about the higher level, and resolve some of the seemingly arbitrary design choices for the lower level. These kinds of insight can also be gained about real systems as well as computational ones.

The high-level entities are symbols. On the lower level, each of these high level symbols is expressed as a collection of lower-level components. For example, the decomposition of hydrogen peroxide into water and oxygen can be written as:

$$2\ \text{hydrogen-peroxide} \rightarrow 2\ \text{water} + \text{oxygen} \qquad (1)$$

But the same equation can be written in terms of the lower level of atoms, instead of in terms of the higher level of molecules:

$$2\,H_2O_2 \rightarrow 2\,H_2O + O_2 \tag{2}$$

Here, 'hydrogen-peroxide' is a symbol on the higher level, that is expressed as two 'H' components and two 'O' components on the lower level. Note also the constraint: 'oxygen' and 'hydrogen-peroxide' are different symbols on the higher level, but they share common components on the lower level: 'hydrogen-peroxide' has the same components as 'oxygen', along with some other components.

On the high level, information about the system is contained in the reaction equations. On the low level, it is contained in the structure of the chemicals (how their components are arranged). So the task of describing a high-level system on a lower level is about *moving information from reaction equations to chemical structures.*

This movement can be performed by humans looking at reaction equations and diagrams. But as the lists of equations become longer and the number of different symbols increases, the problem becomes harder and more tedious to solve. Also, if evolutionary algorithms are to evolve symbolic systems, then this problem needs to be solved hundreds of times for each generation of the evolutionary algorithm. This is why it is useful to have an algorithm for automatically performing this process.

Conservation of mass

The above reasoning relied on the assumption that 'mass' is conserved in the high-level reaction equations: if $\alpha+\beta \rightarrow \gamma$, then all the low-level components making up α and β are present in γ, and γ contains no new components that have not come from α or β.

This is not a difficult condition to fulfill on the high level, as new symbols can be introduced to account for any mass gained or lost in a reaction. For example, if $\alpha + \beta \rightarrow \gamma$, but mass is lost ($\gamma$ does not contain all of the components of α and β), then $\alpha + \beta \rightarrow \gamma$ can be replaced by $\alpha + \beta \rightarrow \xi + \gamma$, where the symbol ξ does not appear anywhere else in the system. ξ represents the mass that is lost in the reaction. Likewise, if mass is gained in the reaction (γ contains a component that does not come from α or β), then $\alpha+\beta+\zeta \rightarrow \gamma$ can be used, where ζ represents the mass gained in the reaction. These two patterns can be applied to any reaction. If they are applied at the same time, they can represent reactions in which some components are lost and some are gained.

Given a high-level system of reaction equations that conserve mass, we can deduce constraints on how the high-level symbols are composed of low-level components. We can also put constraints on the possible masses that the symbols can have. Technically, we deduce a partial order on the masses of the symbols, with constraints of the form: 'χ has more mass than ψ'. We can also use this to work out if a system conserves mass or not, so we do not need to know beforehand. If we encounter a contradiction when building the partial order, we have proved the system does not conserve mass. If we can build the partial order with no contradictions, then we have proved the system does conserve mass.

Multiple meanings

Some high-level reaction equations can have more than one interpretation on the lower level. These can be disambiguated by modifying the reaction equations to include intermediate steps. Different disambiguations lead to different low-level constraints for the same high-level system.

3 chemicals or fewer — unambiguous reactions

There are five kinds of reaction equation that have only one interpretation on the lower level: they involve three molecules or fewer.

1. nothing $\rightarrow \alpha$ (influx)
2. $\alpha \rightarrow$ nothing (outflux, or decay)
3. $\alpha \rightarrow \beta$ (isomerisation)
4. $\alpha + \beta \rightarrow \gamma$ (composition or association)
5. $\gamma \rightarrow \alpha + \beta$ (decomposition or dissociation)

Reaction types (1) and (2) give no information about the lower level (other than saying "α is a symbol that exists"), so are ignored in later analysis.

4 chemicals

Four chemicals participating in a reaction can have more than one interpretation on the lower level.

$3 \rightarrow 1$ reactions. Reactions of the form $\alpha+\beta+\gamma \rightarrow \delta$ imply that chemical δ is a composite of chemicals α, β and γ. The ambiguity lies in the order in which α, β and γ combine to form δ. Because the probability of three molecules reacting with each other at the same instant is negligibly small, two of α, β and γ must react first, the other one reacting with the intermediate complex, ξ. There are three possibilities:

$$\alpha + \beta \rightarrow \xi\,; \quad \xi + \gamma \rightarrow \delta \tag{3}$$
$$\alpha + \gamma \rightarrow \xi\,; \quad \xi + \beta \rightarrow \delta \tag{4}$$
$$\beta + \gamma \rightarrow \xi\,; \quad \xi + \alpha \rightarrow \delta \tag{5}$$

If $\alpha + \beta + \gamma \rightarrow \delta$ were the only reaction in the system, then these three disambiguations would be equivalent. But if α, β and γ participate in other reactions, then the order in which they combine to form δ could have implications on the lower level.

$1 \to 3$ reactions. Similarly to the $3 \to 1$ reactions, reactions of the form $\alpha \to \beta+\gamma+\delta$ can also have multiple interpretations. The chemical α must be composed of chemicals β, γ and δ, and so it must be composed of their low-level components, held together in a certain structure. It must release one of chemicals β, γ and δ first, which implies the existence of an intermediate chemical, ξ, that is the combination of two of β, γ and δ. There are three possibilities:

$$\alpha \to \beta + \xi\,; \quad \xi \to \gamma + \delta \tag{6}$$
$$\alpha \to \gamma + \xi\,; \quad \xi \to \beta + \delta \tag{7}$$
$$\alpha \to \delta + \xi\,; \quad \xi \to \beta + \gamma \tag{8}$$

$2 \to 2$ reactions. Reactions of the form $\alpha+\beta \to \gamma+\delta$ can have multiple interpretations, but these interpretations are of a different kind from those above. The earlier interpretations are about the order in which three chemicals come together to form a complex (or come apart from a complex). The interpretations for $\alpha+\beta \to \gamma+\delta$ reactions concern symbols being transformed into other symbols, which corresponds, on the lower level, to chemicals undergoing isomerisations. There are three possibilities for how this isomerisation can occur:

1. α is an isomer of γ; and β is an isomer of δ.
2. α is an isomer of δ; and β is an isomer of γ.
3. Both α and β contain some components of γ and δ.

Depending on precisely how the lower level will be implemented, point (3) may or may not be possible.

For the purpose of reducing every ambiguous reaction to unambiguous reactions, the reaction $\alpha + \beta \to \gamma + \delta$ can be replaced with the two reactions:

$$\alpha + \beta \to \xi\,; \quad \xi \to \gamma + \delta \tag{9}$$

This again introduces an intermediate complex, ξ. Replacing the equation in this way does not remove the underlying ambiguity. We must make another disambiguation by choosing one of the three cases above.

More than 4 chemicals

In the same way that reactions involving four chemicals can be reduced to unambiguous reactions involving three chemicals or fewer, reactions with more than 4 chemicals can be reduced to unambiguous reactions by the repeated application of the above reductions.

Disambiguation

The first step in the analysis of a high-level system is to pre-process the reactions, reducing them to unambiguous reactions. This involves making choices about how to decompose ambiguous reactions, as described above. If only one ambiguous reaction needs to be decomposed, then the choice made is somewhat arbitrary. But if multiple choices need to be made, then there is the possibility that choices can affect each other.

There is no way in which choices can be incompatible with each other: any set of choices will always lead to a valid disambiguation, and every disambiguation can always be reversed (by removing the intermediates) to return to the same set of ambiguous equations. However, different disambiguations of the same equations can differ in the number of intermediates introduced. If two reactions need to be disambiguated, then this will introduce two new intermediate symbols (one for each reaction). These intermediates are different symbols on the high level, but if there is extra information in the system about the reactants and products of the ambiguous reactions, then it may be possible to relate the intermediates on the lower level, seeing them as isomers of each other (i.e. realising they are composed of the same components). If, however, the equations were disambiguated using different choices, then it might not be possible to relate the intermediates on the lower level. This can also carry over to some of the non-intermediate symbols as well. One disambiguation may make it possible to infer that two non-intermediate symbols are isomers of each other, but a different disambiguation may not make it possible to infer this.

Note that this is not a mistake in the disambiguation process: it is a choice that must be made about how to interpret the high-level equations. If an equation is ambiguous about how one reaction happens, then this ambiguity can carry over to other parts of the system. If application-specific information is available about how ambiguous equations should be disambiguated, then they can be disambiguated by hand before running the analysis. Or if the equations are being generated by a computer program, then this program can be instructed to produce unambiguous equations of the correct form. If it is not known which way the equations would be best disambiguated, then any disambiguation will give a valid representation of the equations. If there is reason to believe that one representation will be better than the others, but it is not known which, then all disambiguations can be enumerated. The analysis can be run on all disambiguations and the results compared to see if multiple representations are possible. If the most compact representation is desired (i.e. the representation that sees the greatest number of symbols as isomers of each other), then this can be found by comparing the different representations. The fact that multiple representations are possible via different disambiguations, highlights the fact that the lower level contains more information than the higher level. Thus we can not map directly to a low-level description from a high-level description; we can only obtain constraints on the lower level.

Algorithm

Once the high-level set of reaction equations has been disambiguated, they can be reasoned about to obtain constraints on the low-level implementation of the system. This reasoning will give us:

- L : a list of low-level components.
- H : a list of the high-level symbols and how they are composed of low-level components.
- I : a list which high-level symbols are isomers of each other.
- P : a partial order on the masses of the components and symbols.

The low-level components here represent constraints on how the lower level must be implemented. These components are no more than the high-level symbols that do not need to be broken down into other symbols. (If a high-level symbol does not need to be broken down on the lower-level, it does not mean that it must not be broken down; it just means there is no information in the high-level reaction equations requiring it to be broken down.)

A set of high-level reaction equations can be thought of as an implicit description of how some symbols in the system are composed of other symbols. The purpose of this algorithm is to make this implicit description explicit. This uses a form of unification [2]. The word 'unification' has a specific meaning in Computer Science (that applies here), but it can be thought of more generally as a way of taking information that is implicit and spread out; making it explicit and bringing it into one place. In this situation, the information is implicitly spread throughout the high-level reaction equations. We are bringing it into an explicit description of how the high-level symbols are composed of low-level components. Off-the-shelf unification algorithms are not suited to this particular situation, as here there is only one function (composition), and it is commutative. So we have designed a special-purpose unification algorithm (algorithms 1 and 2) to exploit the structure of this problem.

Algorithm 1 — set-up

Before we can perform the unification, we need some equations to unify. These will be of the form $\alpha = \beta + \gamma$, representing the fact that the high-level symbol α is composed of the same low-level components as a β symbol combined with a γ symbol. These equations are stored in the data structure D. After the pre-processing steps of disambiguation and removal of influx and outflux reactions, we have isomerisation, composition and decomposition reactions. Algorithm 1 processes these reactions, putting their information into the data structures D, I and P. The decomposition reactions are added as-is into D; the composition reactions are reversed, and added to D. The isomerisation reactions do not need to be put into D, instead their information can be put directly into I. As each equation is added, its information about the partial order on the masses is added to P. When every equation has been processed, the unification can begin. We check the partial order to see if the system conserves mass, and stop now if it does not (because the unification would fail). If there is a contradiction in the partial order, then the high-level system does not conserve mass. If there is not a contradiction then this does not necessarily mean that the system does conserve mass; there is another conservation of mass check during the unification.

Algorithm 1 The first half of the 'downwards' algorithm: Setting up the decompositions of symbols.

```
P := ∅ {partial order on the masses}
I := ∅ {high-level symbols that are isomers}
D := ∅ {decompositions being unified}
for all reaction in high-level-reactions do
  if reaction is α → β {isomerisation} then
    add isomer 'α = β' to I
    add order relation 'α = β' to P
  else if reaction is α + β → γ {composition}
       or γ → α + β {decomposition} then
    add decomposition 'γ = α + β' to D
    add order relations 'α < γ' and 'β < γ' to P
  if there is a contradiction in P then
    return failure: the system does not conserve mass
```

Algorithm 2 — unification

After set-up stage, the data structure D is filled with the equations to unify. Algorithm 2 performs this unification and completes the 'downwards' algorithm. D contains a list of equations of the form $\omega = \chi + \psi$, where ω, χ and ψ are symbols from the high-level system (or intermediates generated by disambiguation). The equation $\omega = \chi + \psi$ means that the symbol ω is composed of the same low-level components as the symbols χ and ψ. But D could contain another equation: $\omega = \tau + \upsilon$. These two equations both describe how ω is composed, and need to be considered together during this step of the algorithm. During this step we iterate through the equations in D, grouping together all equations describing the same symbol (e.g. ω). So in a typical iteration we might consider the decompositions $d = d_1 = d_2$, where d is ω, d_1 is $\chi + \psi$ and d_2 is $\tau + \upsilon$. So the notation $d = d_1 = d_2$ means that we are considering the two equations, $\omega = \chi + \psi$ and $\omega = \tau + \upsilon$.

For each of these sets of decompositions, we apply one of five operations (in order) to simplify the equations. This process is iterated until no equations remain. Then the equations have been unified and the process is complete. WHen simplifying the equations, we may find a way to partially decompose a symbol but not know its full decomposition yet. This information is stored in the temporary variable PA,

which is like H but stores partial information about decompositions. The operations that we perform are:

1. If a symbol has only one decomposition ($d = d_1$) then this symbol has been fully decomposed. We add this decomposition to H, including any partial decomposition already done to d.

2. If there are common symbols in the decompositions of a symbol ($d = \chi + \psi = \tau + \psi$) then we cancel these and add them to the partial decomposition of d.

3. If any decompositions of a symbol contain only one symbol themselves, then we can cancel them. We remove all but one of these decompositions from D, and add into I the fact that these symbols are all isomers of each other. We also update the partial order, P, with the fact that these symbols all have the same mass (and we check the partial order for contradictions).

4. Because different symbols can be isomers of each other, we replace all instances of these isomers with a common identifier so they can be cancelled from the equations by operation 2.

5. If none of the above operations can be performed, then we search the decompositions for the first symbol that we know how to decompose (it has an entry in H). We replace this symbol in its equation by its decomposition. So if $\omega = \psi + \chi = \tau + \upsilon$ and $\tau = \rho + \sigma$ then we end up with $\omega = \psi + \chi = \rho + \sigma + \upsilon$.

After all the equations have been unified, the set of low-level components, L, can be read off as those high-level symbols that can not be decomposed (are not in H) and are not isomers of a different high-level symbol (are not in I).

Algorithm 2 The second half of the 'downwards' algorithm: Unifying the decompositions

```
L := ∅ {low-level components}
H := ∅ {high-level symbols to be broken down}
PA := ∅ {partial decompositions}
while D is not empty do
  for all d = d_1 = d_2 = ··· = d_n in D do
    if n = 1 then
      add decomposition 'd = PA(d) ∪ d_1' to H
      remove decomposition 'd = d_1' from D
    else if common symbols in d_1 = d_2 = ··· = d_n
    then
      cancel the common symbols
      add the common symbols to PA(d)
    else if more than one of d_1 = d_2 = ··· = d_n are
    length 1 then
      s_1 = s_2 = ··· = s_m are these decompositions
      for all unique pairs s_i, s_j do
        add isomer 's_i = s_j' to I
        add order relation 's_i = s_j' to P
        if there is a contradiction in P then
          return failure: system not conserve mass
      remove all but one of s_1 = ··· = s_m from D
    else if at least one of d_1 = d_2 = ··· = d_n contains a
    chemical in S then
      for all matching chemicals c do
        replace c with its common identifier from I
    else
      find the first d_i in d_1 = d_2 = ··· = d_n with a
      match in H
      replace 'd = d_i' in D with 'd = H(d_i)'
L := {all high-level symbols} \ (H ∪ I)
return success: L, H, I, P
```

Going upwards

Moving up from a low-level system to a high-level system is more straightforward than moving down.

The precise implementation details of the lower level system do not matter for the process of moving up to the higher level. However the low-level system is implemented, it will consist of components that interact with each other and join together to form structures. (For example, two hydrogen atoms and one oxygen atom may join to form a water molecule structure.) These structures are symbols on the higher level. The reactions on the higher level summarise the low-level mechanisms by which these structures interact. To produce a high-level description of a low-level system, two things are needed: (1) a list of high-level symbols; and (2) a list of reactions involving these symbols. The symbols represent the structures formed by the low-level components and the reactions represent the dynamics happening on the lower level. Algorithm 3 gives the pseudocode of an algorithm to do this.

To produce a list of symbols, it is necessary to simulate the low-level system and observe the structures that form. The length of time the system is observed for has an impact on the structures observed. If very involved structures could form within the system but they take longer to form than the system is simulated for, then they will not be observed. Likewise if some structures form quickly but rarely, they may not be observed if the system is not simulated for long enough. This highlights the fact that the high-level system is an approximation of the low-level system, capturing those structures that form within a certain timescale.

There is another timescale associated with the observation of the low-level system. When observing structures within the system, a short timescale must also be chosen. Because the low-level components are constantly interacting with each other, a complicated structure goes through intermediate stages in its formation. These intermediate stages may not be appropriate to represent in the high-level system: the only thing required may be the resulting structure. The

Algorithm 3 Going upwards from a low-level description to a high-level description of a system.

```
S := ∅ {set of high-level symbols}
R := ∅ {set of high-level reactions}
while long timescale has not expired do
  while short timescale has not expired do
    simulate low-level system
  observe low-level system
  for all new structures not seen before do
    create a new symbol for this structure
    add this new symbol to S
  for all structures, S, at the start of this timescale do
    if S was in a reaction during this timescale then
      A := { structures that S reacted with }
      B := { products remaining after these reactions }
      create a new symbolic reaction: A → B
      add this reaction to R
```

separate, simple operations happening on the low level are combined into one complicated operation on the high level. This again highlights the fact that the high-level system is an approximation of the low-level system. A complicated structure that forms through intermediate stages on the lower level springs into being in one step on the higher level.

Observation of the low-level system gives a list of reaction equations as well as a list of symbols. The short timescale is used to approximate a series of intermediate structures by one symbol: the end product of the series. This approximation gives a reaction equation. Whatever structures were present in the area of interest at the start of the short timescale are the reactants in the reaction equation, and whatever structures were left over after the short timescale are the products of the reaction equation. Thus the observation of symbols also gives a list of reaction equations. For a new symbol to be observed, there must have been a process taking place by which the symbol was formed. This process is observed and approximated by the short timescale. This gives a new symbol (or symbols), and a reaction creating the symbol(s). Repeating this observation of the low-level system for the whole duration of the long timescale gives a list of high-level symbols and a list of reaction equations. This is a high-level description of the system.

Conclusions and future work

Building AChems on multiple levels provides more flexibility than using just one level. It may provide a way of approaching difficult problems in the evolution of complexity, such as the 'major transitions in evolution' [14]. This paper presents some initial thoughts about moving between levels, and some algorithms that allow systems to automatically move between levels.

An algorithm is presented for moving down from a high-level description of a system to a lower level of description. Conservation of mass is needed in the high-level system in order to infer information about the low-level system. The algorithm can be used to determine if a system conserves mass or not. If the system does conserve mass, then the analysis can be performed. If it does not, then the algorithm can be used to determine precisely which parts of the system do not conserve mass. Since a high-level description is an approximation of a low-level system, this algorithm generates a set of constraints that any low-level implementation of the system must satisfy. Depending on the precise way in which the low-level system is implemented, this either provides a way of generating an implementation, or it provides a criterion that can be used to search for an implementation.

We will use this algorithm to investigate different low-level implementations of AChems. We have developed some implementations where the constraints generated by this algorithm map directly into low-level descriptions. We also have some sub-symbolic representations [5] where these constraints can be used to search for sub-symbolic chemistries that fulfil the high-level description.

Some high-level systems are ambiguous as to how they operate on the low-level. This algorithm can be used on different disambiguations of the high-level system to give information about the system. We will build a tool to show which parts of the system are most ambiguous, and which are most constrained on the lower level. This information may be helpful, particularly in guiding algorithms that are searching for low-level implementations.

The algorithm introduces intermediate chemicals into the system to disambiguate reactions. A consequence is that reactions happening in one step in the high-level system can take multiple steps to happen in the low-level system. Intermediates can interact with other parts of the system, disrupting the reaction. Things not possible in the high-level system become possible by moving to a lower level of description. So some richness is added into the system by a low-level description, which may be useful to other processes that are exploiting the AChem. For example, the extra richness can provide more ways in which to evolve the reactions.

There is a further part to the 'downwards' algorithm, which we will develop. As well as knowing how high-level symbols are composed of low-level components, it would be useful to know precisely how these low-level components are connected together. If we consider the low-level components connected to each other by 'binding sites', then we can work this out. Each binding site has an affinity to each other binding site, and each component can have many binding sites. Components binding to each other can cause new binding sites on the components to become available, and existing ones to become unavailable. After running the presented 'downwards' algorithm, we have enough information to work out how many binding sites each low-level component needs to have, and which sites must be able to bind with which others. If the high-level reaction equations come with

reaction rates, then in principle we could carry these rates through the algorithm and work out values for the affinities on the binding sites (although this would require knowing how the kinetics will be simulated on the lower level). The concept of binding sites shows further richness gained by using a low-level description of a system. Rather than listing which components must come together to form which high-level symbols, we just list which binding sites each component must possess. The high-level symbols come out from the low-level system as a consequence of the binding sites possessed by each component in the system. Creating a new component only involves creating an arrangement of binding sites (with affinities to sites already in the system). Adding a new component to an existing system changes the high-level structures the system can form.

An algorithm is also presented for moving up from a low-level description to a high-level description. A high-level description is described as an approximation of the low-level description, and this approximation is made precise by the description of two different timescales that constitute this approximation. A short timescale is used to approximate the interactions and intermediate structures on the lower level into symbols and reactions on the higher level. A long timescale is chosen to give a period over which the low-level system will be observed, and only those events occurring within this time period will be approximated.

We will link the two algorithms presented here. One way to do this is with a heuristic search algorithm operating on two different levels. A search algorithm (such as an evolutionary algorithm) is used to search for an AChem to solve a particular problem. A common issue encountered when designing heuristic search algorithms is which problem representation to choose. This issue can be somewhat avoided by representing solutions to the problem as two-level AChems. The search algorithm can search through different high-level representations of the AChem until it becomes stuck in a local optimum. It can then switch to the low-level representation and search in this representation for a time (perhaps until it becomes stuck in another local optimum). Now, it can move back to the high-level representation. When it does this it will not only find itself in a different part of the high-level search space, but it may find itself in a different high-level search space altogether. Because the low-level representation can easily create new high-level symbols, moving down to the low-level description and running the search will change the symbols that exist on the high-level, and change the relationships of existing symbols to each other. Likewise, running a search on the high level and moving back down to the low level has the potential to change the type of low-level representation that will be generated by the 'downwards' algorithm. This searching on two levels effectively co-evolves two different problem representations. It is just one way in which the two tools provided by this paper can be used.

Acknowledgements

The authors would like to thank the other members of the Plazzmid project at the University of York for valuable comments and ideas throughout the work: Ed Clark, Simon Hickinbotham, Peter Young, Tim Clarke and Mungo Pay.

This work is part of the Plazzmid project, funded by EPSRC grant EP/F031033/1

References

[1] Anton Arkhipov, Peter L. Freddolino, and Klaus Schulten. Stability and dynamics of virus capsids described by coarse-grained modeling. *Structure*, 14(12):1767–1777, 2006.

[2] F. Baader and W. Snyder. Unification theory. In J. A. Robinson and A. Voronkov, editors, *Handbook of Automated Reasoning*, volume 1, pages 447–533. Elsevier Science, 2001.

[3] M. Conrad. Cross-scale interactions in biomolecular information processing. *Biosystems*, 35(2-3):157–160, 1995.

[4] P. Dittrich, J. Ziegler, and W. Banzhaf. Artificial chemistries–a review. *Artificial Life*, 7(3):225–275, 2001.

[5] Adam Faulconbridge, Susan Stepney, Julian Miller, and Leo Caves. RBNWorld: Sub-symbolic artificial chemistry. In *ECAL 2009, Budapest, Hungary, September 2009*. LNCS. Springer, September 2009.

[6] Verena Fischer and Simon Hickinbotham. A metabolic subsumption architecture for cooperative control of the e-puck. In *To appear in: International Workshop on Nature Inspired Cooperative Strategies for Optimization (NICSO 2010)*. Springer, 2010.

[7] W. Fontana. Algorithmic chemistry. *Artificial Life II*, pages 159–210, 1992.

[8] Simon Hickinbotham, Edward Clark, Susan Stepney, Tim Clarke, Adam Nellis, Mungo Pay, and Peter Young. Molecular microprograms. In *ECAL 2009, Budapest, Hungary, September 2009*. LNCS. Springer, September 2009.

[9] Simon Hickinbotham, Edward Clark, Susan Stepney, Tim Clarke, and Peter Young. Gene regulation in a particle metabolome. In *CEC 2009, Trondheim, Norway, May 2009*. IEEE Press, May 2009.

[10] W. S. Hlavacek, J. R. Faeder, M. L. Blinov, R. G. Posner, M. Hucka, and W. Fontana. Rules for modeling signal-transduction systems. *Science's STKE : signal transduction knowledge environment*, 2006(344), 2006.

[11] P. J. Hoogerbrugge and J. M. V. A. Koelman. Simulating microscopic hydrodynamic phenomena with dissipative particle dynamics. *EPL (Europhysics Letters)*, 19(3):155–160, 1992.

[12] Tim J. Hutton. Evolvable self-replicating molecules in an artificial chemistry. *Artificial Life*, 8(4):341–356, October 2002.

[13] Duraid Madina, Naoaki Ono, and Takashi Ikegami. Cellular evolution in a 3D lattice artificial chemistry. In *Advances in Artificial Life*, pages 59–68. 2003.

[14] John Maynard Smith and Eörs Szathmáry. *The major transitions in evolution*. Perseus Books, 1999.