Organic Synthesis as Artificial Intelligence Planning

Arman Masoumi[1], Mikhail Soutchanski[1], and Andrea Marrella[2]

[1] Ryerson University, Toronto, Ontario, Canada
arman.masoumi@ryerson.ca, mes@scs.ryerson.ca
[2] Sapienza Università di Roma, Rome, Italy
marrella@dis.uniroma1.it

Abstract. We explore advantages that can be gained from using expressive logic languages for semantic modelling of chemical reactions. First, we present a novel approach for logical representation of notions in organic chemistry, as well as for reasoning about generic chemical reactions. Subsequently, using this new semantic modeling of reactions, we explore what reasoning problems can be solved. We focus on solving organic chemistry synthesis problems, where the goal is to synthesize the target molecule from a set of starting stage molecules. We argue that this problem can be reduced to a planning problem in Artificial Intelligence. We conduct experimental study including empirical assessment of a PROLOG planner and two state-of-the-art planners. We investigate if they are capable of solving a set of instances of the organic synthesis problem. We report numerical data from our study and do comparative analysis of the planners. The novelty of our work is in using state-of-the art planners for solving the organic synthesis problem. The significance of our work is in methodology that we developed and in showing that expressive logical language can be useful for semantic modeling.

Keywords: Organic Chemistry, Computer-Assisted Organic Synthesis, Situation Calculus, Planning, PDDL, Knowledge Representation and Reasoning, Datalog.

1 Introduction

Recently, there is a growing interest in using semantic technologies for the purposes of unambiguous representation of chemical knowledge about small molecules [43,11,41]. A number of ontologies are being developed to facilitate annotation, sharing of chemical data about molecules and to guarantee interoperability of tools processing chemical data and knowledge [12,13,33,35]. For example, the CHEBI ontology aims to represent all Chemical Entities of Biological Interest [19,34]. In a similar vein, there is interest in bioinformatics [3,20] towards standardized representation (e.g., in *BioPAX*) of biological pathways, which include sequences of bio-chemical reactions in metabolic pathways. However, it becomes increasingly evident that further progress of research on developing useful ontologies in cheminformatics and bioinformatics is limited by expressiveness of the logical languages, such as OWL2 [40,30], which are commonly employed to formulate the ontologies. For example, the current version of CHEBI incorporates only isA relations between molecular entities and a bit of mereology to show parthood relations between molecules and the constituent functional groups, but there is no complete representation of structure of the molecules. As argued in [35,49,50,53], there is a growing need in considering more expressive logical languages, because they are needed to formulate complex concepts, such as molecules with rings, that cannot be correctly represented in OWL2 alone. Modeling and reasoning about biochemical reactions and pathways in OWL2 is also limited [22].

In the area of formal ontologies, it is recognized that actions, events and other dynamic entities are better understood as independent entities, e.g., see Basic Formal Ontology (BFO) [59,60,5], Descriptive Ontology for Linguistic and Cognitive Engineering (DOLCE) [26] and General Formal Ontology (GFO) [39,4]. In a somewhat similar vein,

[54] argues that actions should be reified into first class entities that can be quantified over. In addition, in philosophy of action, Donald Davidson [18] argued that the logical form of sentences in which actions (or events) are adverbially modified can be stated only when the adverbs are transformed into predicates about an event, and the form is understood as an implicit quantification over the event applied to the conjunction of these predicates that have the event as an argument. Moreover, Barry Smith coined the term "fantology" to criticize a belief that the ontological structure of reality can be captured exclusively using unary and binary predicates [59,60].

In the long-term perspective, we would like to investigate what can be gained from using expressive logics and from integrating static ontologies (e.g., formulated in OWL2) with a more traditional Artificial Intelligence (AI) approach to reasoning about actions. It is important to design a hybrid logical representation that balances expressivity with computational tractability of reasoning and that allows to solve a variety of problems including both classification of static entities and goal-directed planning over actions. Some initial steps in this research direction are taken in [31,67,7,56,55]. In this paper, we concentrate on exploring advantages that can be obtained from using expressive logical languages for representing biochemical reactions. It remains to be seen how our representation of reactions can be integrated with existing chemical ontologies of static chemical entities (e.g., ontologies mentioned above). More specifically, in this paper, we show that by using an expressive logical language, one can easily formulate and solve the planning (synthesis) problems in the area of organic chemistry.

The organic synthesis problem can be understood as identifying a sequence of chemical reactions that can transform a set of starting state molecules into the target molecule. This problem is of great importance since it has pharmaceutical and industrial applications. In an effort to facilitate discovery in life sciences, and more specifically in organic chemistry, a novel way of representing and reasoning about chemical reactions was introduced in [51]. The proposed approach can represent generic chemical reactions and reason about them at the level of changing chemical bonds between the atoms. The introduced approach is based on the *Situation Calculus* (SC), a well-known logical AI formalism for representing and reasoning about dynamic domains. The SC reasoning procedures have been extensively studied in the AI community, and thus it provides a significant potential for modeling of dynamic phenomena in life and natural sciences. Following the approach proposed in [51], we formulate the organic synthesis problem as a planning problem in AI and explore whether existing planner systems are capable of solving the problem. We investigate two different computational approaches to solving this kind of planning problems and do an empirical assessment of these approaches on a set of benchmark problems. We start with encoding instances of the organic synthesis in two different languages. One of the planners (that we have developed ourselves) solves the planning problems directly in SC and accepts PROLOG encoded input. This planner can take advantage of domain specific declarative heuristics built ad-hoc for helping the system in synthesizing a goal molecule. The other planners that we used in our experimental study are the state-of-the-art planners that solve the planning problems using a different methodology. They are the winners of the International Planning Competitions. These state-of-the-art planners support a standard input language called the Planning Domain Definition Language (PDDL). The PDDL planners use domain-independent heuristics. To facilitate evaluation, we translated manually our small library of generic reactions from PROLOG to PDDL.

We collect run-time data from each of these planners and compare their performances. Our experimental study is preliminary and more broad assessment is required on a much greater variety of instances of the organic synthesis problem. However, the main novelty of our paper is that studies of this kind can be proposed and conducted. To the best of our knowledge, the PDDL state-of-the-art planners have not been previously used to solve organic synthesis problems. The significance of our empirical assessment is that it indicates what new research directions should be taken following this work.

2 Background

In what follows, all free variables (typically starting with lower case letters such as object variables x, situation variable s, and a variable of sort action a) are implicitly universally ($\forall$) quantified at front. Read the symbol $\wedge$ as 'and', $\vee$ as 'or', $\exists$ as 'exists', $\neg$ as 'not'. We use notation $\overrightarrow{x}$ to denote a tuple of object arguments. A number of examples are used in this section to clarify the concepts being introduced. The examples are inspired by an application domain that was used in the International Planning Competition (IPC) in 2005, namely the *Pathways* domain [28]. Later, when we explain our approach to representing chemical reactions, we will point out the differences and benefits of our approach over the Pathways domain.

The situation calculus (SC) is a logic formalism designed for representing and reasoning about dynamical domains. Basic ingredients of SC are *actions* a, *situations* s and *fluents* [58]. In SC, *actions* are used to represent changes occurring in an application domain. For example, if the domain is molecular biology, the action $associate(m_1, m_2, m_3)$ can represent a biochemical association reaction where molecules m_1 and m_2 react (and are consumed) to form the new complex molecule m_3. There are usually a number of objects in a domain as well. For example, in the area of molecular biology, the objects (constants) of the domain would be specific molecules or proteins, such as protein $PCAF$. Performing actions results in changing the *situation*. A situation is a first-order term denoting a sequence of actions. The special constant S_0 denotes the *initial situation*, namely the empty action sequence. The function $do(a, s)$ denotes the new situation that results from performing action a in situation s. For example the situation where proteins $PCAF$ and $P300$ reacted to produce protein $PCAF$-$P300$ (and nothing else has happened) can be represented by $do(associate(PCAF, P300, PCAF\text{-}P300), S_0)$. *Fluents* are predicates whose values may vary from situation to situation, and therefore are predicates with the last argument s being a situation. They generally describe those features of the application domain that may change when actions are executed. As an example, consider the fluent $Available(m_1, s)$ that holds in situation s if the molecule m_1 is available in s.

A basic action theory (BAT) D is a set of axioms in SC that is used to model actions, their preconditions, their direct effects and initial values of the fluents.

Initial theory D_{S_0}: a set of axioms representing the initial situation, before any action was performed. In the initial theory, we might have also sentences with no situation argument in them; they may include external static ontologies or facts that do not change. For example in the initial theory we might assert that there is a biochemical association reaction in which the proteins $PCAF$ and $P300$ react to produce $PCAF$-$P300$ using the predicate $Association\text{-}reaction(PCAF, P300, PCAF\text{-}P300)$. The initial theory might be logically incomplete if not all the facts about an environment are known.

Action precondition axioms D_{ap}: a set of axioms characterizing possibility of executing an action. Precondition axioms (PAs) use the distinguished predicate $Poss(A(\overrightarrow{x}), s)$ meaning that an action $A(\overrightarrow{x})$ is possible in situation s. (Recall that $\overrightarrow{x}$ represents a tuple of arguments.) There is one axiom for each action term $A(\overrightarrow{x})$, with syntax $Poss(A(\overrightarrow{x}), s) \leftrightarrow \Pi_A(\overrightarrow{x}, s)$. $\Pi_A(\overrightarrow{x}, s)$ represents the preconditions of action A: A is possible if and only if (use the bi-conditional $\leftrightarrow$ for iff) the logical condition $\Pi_A(\overrightarrow{x}, s)$ holds in s. In the example that we have been following, a possible precondition axiom could be:

$Poss(associate(m_1, m_2, m_3), s) \leftrightarrow$

$Available(m_1, s) \wedge Available(m_2, s) \wedge Association\text{-}reaction(m_1, m_2, m_3)$.

This PA is stating that the biochemical association reaction between molecules m_1 and m_2 to produce m_3 is possible only if both m_1 and m_2 are available in situation s, and there is a biochemical association reaction transforming m_1 and m_2 into m_3.

Successor state axioms D_{ss}: a set of axioms characterizing the effects of the action on the fluents. The idea behind successor state axioms is that a fluent becomes true after

executing an action if the action causes it to become true, or the fluent remains true if it was already true and the action taken did not cause it to become false. But otherwise, the fluent becomes false, if the most recently executed action has a negative effect on the fluent. More formally speaking, each SSA has the following generic form:

$$F(\overrightarrow{x}, do(a,s)) \leftrightarrow \bigvee_i a = \textit{PosAction}_i(\overrightarrow{x}) \wedge \gamma_i^+(\overrightarrow{x}, s) \vee F(\overrightarrow{x}, s) \wedge \neg\big(\bigvee_j a = \textit{NegAction}_j(\overrightarrow{x}) \wedge \gamma_j^-(\overrightarrow{x}, s)\big),$$

where $\textit{PosAction}_i$ is an action that makes the fluent F true and $\gamma_i^+(\overrightarrow{x}, s)$ is the formula expressing a context in which this positive effect can occur; similarly, $\textit{NegAction}_j$ is an action that can make the fluent F false if the context formula $\gamma_j^-(\overrightarrow{x}, s)$ holds in s. If the executed action a is none of these, then the truth value of F remains unchanged (a has no effect). SSAs characterize the truth values of the fluent F in the next situation $do(a, s)$ in terms of fluents in the situation s and they represent non-effects of actions compactly (because of implicit universal quantifier $\forall$ over action variable a). For example, the successor state axiom for the fluent *Available* is as follows:

$$\begin{aligned} Available(m, do(a,s) \leftrightarrow \exists m_1, m_2 (a = associate(m_1, m_2, m)) \vee \\ Available(m,s) \wedge \neg\exists m_1, m_2 (a = associate(m, m_1, m_2) \vee \\ a = associate(m_1, m, m_2)). \end{aligned}$$

This SSA is stating that a molecule m becomes available, if the most recently executed action is $associate$ with the last argument m (recall that last argument of $associate$ is the molecule that is produced as the result of the reaction), or the molecule m was already available in s, and the last action did not consume it, i.e., the last action was not $associate$ with m appearing as either first or second arguments.

BATs might also be augmented with *abbreviations*, also known as *derived predicates* [61]. Abbreviations look similar to fluents in the sense that they are also predicates with their last argument a situation. Similar to fluents, their truth value can vary from situation to situation. However, abbreviations differ from fluents in that the actions do not directly affect them. Instead, the effects of the actions on abbreviations are implicit. So, there are no SSAs for abbreviations. Sometimes, abbreviations can be eliminated, but it is convenient to keep them to make other axioms more succinct. In addition, axioms defining abbreviations help to make a logical theory more modular. For example, abbreviations can occur in the precondition axioms and can represent common terms of an application domain. Syntactically, axioms defining abbreviations are formulas *uniform* in s: these are formulas that have only occurrences of fluents and/or abbreviations with the situation argument s, may have also occurrences of static predicates, but cannot mention any other situation variables, terms or quantifiers over situations. Abbreviations may have arbitrary many object arguments that usually represent key atoms in the functional groups which are part of the molecules. As an example, consider the hypothetical abbreviation that can be added to the Pathways domain:

$$Dangerous(m, s) \leftarrow Molecule(m) \wedge Available(m, s) \wedge Available(PCAF, s).$$

Here, for the sake of an example, we are assuming that having any molecule m is dangerous if protein $PCAF$ is also available. Notice that there is no action that can make the predicate $Dangerous$ true directly. This is why the most intuitive way to have it properly defined is through a state constraint. Since actions have direct effect on the fluent $Available(m, s)$, they also have indirect effect on the property $Dangerous$, e.g., as soon as m becomes available, it is dangerous, but if m is no longer available, it ceases to be dangerous.

Planning Domain Definition Language (PDDL) is the standard language with Lisp-like syntax developed for expressing planning problems [52]. It is used as the input language in bi-annual competitions: all instances of the planning problems must be specified in PDDL. It is worth noting that SC provides declarative semantics for PDDL [14], and it is not difficult to translate from one syntax to the other. In the remainder of this paper, we use the SC syntax.

3 Methodology

In this section we explain how chemical reactions can be represented in SC BATs. This work was originally introduced in [51]. Previously, Fujita proposed a formalism to model chemical reactions called Imaginary Transition Structure (ITS) [25]. A somewhat similar approach was developed by G.E. Vladutz, who proposed "superimposed reaction graphs " and "superimposed reaction skeleton graph" [66,64]. In either case, the approaches are based on graph-transformation modeling of chemical reactions, which are in spirit similar to how SSAs are constructed. The main idea is that, as a result of a reaction, a number of new bonds can be formed, a number of bonds can split, and all remaining bonds will not change. This closely matches the main idea behind the SSAs, where an action causes some fluents to become true, some fluents to become false, and has no effect on others. Of course, SSAs [58] are more general in terms of applicability.

Representing molecules requires representing chemical atoms and the bonds between them. The chemical atoms form the objects in the SC formulas and their type is determined by the situation independent predicates corresponding to the atom names in Mendeleyev's periodic table. For example, $Carbon(C_1)$ means the constant C_1 represents a carbon atom and $Hydrogen(H_1)$ means H_1 represents a hydrogen atom; $Atom(x)$ declares x is any atom in an initial theory. Similarly, the groups in the periodic table can be represented. For example, $Halogen(x)$ is true if x belongs to the Halogen group in the periodic table (e.g., x might be Fluorine or Chlorine). The chemical bonds between the atoms are represented by fluents describing the type of the bond. For example, the fluent $Bond(x, y, s)$ is true iff atom x has a single bond with atom y in situation s and the fluent $DoubleBond(x, y, s)$ is true iff there is a double bond between x and y in situation s; $TripleBond(x, y, s)$, $AromaticBond(x, y, s)$ are similar.

The molecules that are available in the initial situation S_0 are described in the initial theory D_{S_0} by formulas introducing the atoms and the bonds between them. For example, a water molecule H_2O in the initial situation S_0 can be represented as

$$Hydrogen(H') \wedge Hydrogen(H'') \wedge \quad H' \neq H'' \wedge Oxygen(O) \wedge$$
$$\forall x(Bond(x, O, S_0) \rightarrow (x = H' \vee x = H'')) \wedge$$
$$\forall y(Bond(H'', y, S_0) \rightarrow y = O) \wedge \forall y(Bond(H', y, S_0) \rightarrow y = O).$$

Representing generic chemical reactions requires representing and identifying *classes of molecules*. Molecules within the same chemical class have similar chemical characteristics. For example, *alkanes*, *alcohols* and *esters* are some of the well-known chemical classes that display unique behaviors in reactions. Chemical classes owe their chemical characteristics to the *functional groups* constituting them. Functional groups are specific sub-molecules in a molecule. For example, *alkyl*, *hydroxyl* and *ester*, are the main functional groups in alkanes, alcohols and esters, respectively. Alkyls, usually denoted by R, are chemical compounds that consist solely of acyclic single bonded (univalent) carbon and hydrogen atoms with the generic formula C_nH_{2n+1}. For example, *methyl* CH_3- and *ethyl* CH_3-CH_2- are alkyls. A hydroxyl group OH- is an oxygen atom O that is bound with a hydrogen atom H. The ester functional group has the form $COO-R$, where R is an alkyl. Alkanes are compounds in which the alkyl functional group bonds with a hydrogen atom. For example, *methane* CH_4 is an alkane. An alcohol $R-OH$ is a compound in which a hydroxyl functional group -OH is bound to a carbon atom in an alkyl R: for instance, *methanol* CH_3-OH and *ethanol* CH_3-CH_2-OH are alcohols. Esters are compounds of the form $R-COO-R'$, where both R and R' are alkyls, and an oxygen atom has a double bond with carbon.

We use abbreviations to define different chemical classes, functional groups and specific instances of molecules. Abbreviations correspond to derived predicates in PDDL [61]. The arguments of the abbreviations are referred to as *key atoms* of the abbreviation, which are chosen from the atoms at common reaction sites of a given molecule/chemical class. For example, the abbreviation for alcohol $R-OH$ could be:

$$Alcohol(o, h, s) \leftarrow Hydroxyl(o, h, s) \wedge \exists c(Alkyl(c, s) \wedge Bond(o, c, s)),$$

where $Hydroxyl(o,h,s)$ is an abbreviation as follows:
$Hydroxyl(o,h,s) \leftarrow Oxygen(o) \wedge Hydrogen(h) \wedge Bond(o,h,s) \wedge$
$\exists^{=2}\, x(Atom(x) \wedge Bond(o,x,s)) \wedge \exists^{=1}\, x(Atom(x) \wedge Bond(h,x,s))$.
In the above formula, $\exists^{=1}$ ($\exists^{=2}$, respectively) is a counting quantifier saying that there exists exactly one (there are exactly two, respectively) entities for which quantified formula holds. The counting quantifiers can be replaced with usual (but less readable) first order logic syntax. For example, $\exists^{=2}\, x(\varphi(x))$ stands for $\exists x_1 \exists x_2(\varphi(x_1) \wedge \varphi(x_2) \wedge x_1 \neq x_2 \wedge \forall y(\varphi(y) \rightarrow (y = x_1 \vee y = x_2)))$. Notice that in the abbreviation for alcohol we use another abbreviation, namely $Alkyl(c,s)$. The abbreviation for alkyl is written recursively and its key atom is the carbon atom that is not saturated with hydrogen or other carbon atoms, but can form a bond.

As another example, molecules that include chlorine and hydrogen somewhere in their structure can be defined using the following abbreviation:
$ChlAndHydr(cl,h,s) \leftarrow Chlorine(cl) \wedge Hydrogen(h) \wedge Connected(cl,h,s)$.
where $Connected(x,y,s)$ is the recursive transitive closure relation defined on bond fluents. For the sake of simplicity, if there are single bonds only, it is defined as follows:
$Connected(x,y,s) \leftarrow Bond(x,y,s)$.
$Connected(x,y,s) \leftarrow \exists z(Bond(x,z,s) \wedge Connected(z,y,s))$.

Furthermore, chemical concepts such as hydrocarbon, which are compounds consisting entirely from carbon and hydrogen atoms, or inorganic molecules, i.e. molecules that do not contain any carbon atoms can be defined as follows:
$Hydrocarbon(c,h,s) \leftarrow Carbon(c) \wedge Hydrogen(h) \wedge Connected(c,h,s) \wedge$
$\neg\exists x(Atom(x) \wedge Connected(c,x,s) \wedge \neg Hydrogen(x) \wedge \neg Carbon(x))$.
$Inorganic(x,s) \leftarrow Atom(x) \wedge \neg Carbon(x) \wedge \neg\exists c(Carbon(c) \wedge Connected(c,x,s))$.

We provided examples of how functional groups and chemical classes can be represented as abbreviations above. However, we can also easily represent specific molecules, including molecules with rings, e.g., Cyclobutane, which is a cyclic structure constructed from four carbon atoms with single bonds with each other, each of which has two distinct hydrogen atoms attached to. Moreover, abbreviations are also used for representing goal molecules of the planning instance. For example,
$Goal(s) \leftarrow (\exists o,h)Alcohol(o,h,s) \wedge (\exists cl,c)EthylChloride(cl,c,s)$.
This formula states that our goal is to reach a situation s in which there are atoms identifying an alcohol molecule, as well as a molecule of ethyl chloride $CH_3{-}CH_2{-}Cl$.

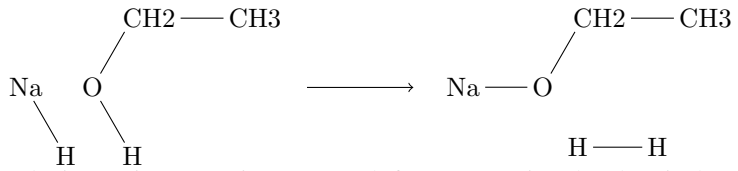
Formally, the bodies of abbreviations are constructed from fluents, abbreviations, conjunctions, existential quantifiers and (restricted) negation. The abbreviations can be expressed as a stratified $Datalog^{\neg}$ program [1,8], possibly after applying the Lloyd-Topor transformations [46]. The Lloyd-Topor transformations are needed to transform the rules that use $\neg\exists, \forall, \vee$ (e.g., due to eliminating counting quantifiers) into logically equivalent $Datalog^{\neg}$ rules, e.g., the transformed rule for $Hydroxyl(o,h,s)$ would be:
$Hydroxyl(o,h,s) \leftarrow Oxygen(o) \wedge Hydrogen(h) \wedge Bond(o,h,s) \wedge$
$\exists^{=2} x(Atom(x) \wedge Bond(o,x,s)) \wedge \exists x_1(Atom(x_1) \wedge Bond(h,x_1,s) \wedge \neg P(h,x_1,s))$,
where $P(h,x_1,s)$ is the auxiliary predicate introduced by one of Lloyd-Topor's transformations. The rule defining $P(h,x_1,s)$ is as follows:
$P(h,x_1,s) \leftarrow \exists y(Atom(y) \wedge Bond(h,y,s) \wedge y \neq x_1)$.
The remaining counting quantifier $\exists^{=2}$ can be similarly eliminated. Note that in the above examples for inorganic and hydrocarbon molecules with ($\neg\exists$), the predicate $Connected$ was defined in previous rules. Using Lloyd-Topor's transformations, these rules can be transformed into stratified $Datalog^{\neg}$. Thanks to stratification of our abbreviations, there is always an unique minimal model satisfying our abbreviations. It is known that data complexity of stratified Datalog$^{\neg}$ is P-complete [8].

Now we are ready to introduce the SC actions for representing chemical reactions and write PAs and SSAs for them. We define an action for each generic chemical reaction, with the arguments of the action being the atoms that change bond in the chemical

reaction. The reactions are generic in the sense that usually they represent reactions between chemical classes rather than specific molecules. For example, a known chemical reaction is the reaction between alcohols and bases. This reaction can be represented as a SC action ($\underline{a}lcohol\_\underline{b}ase\_\underline{r}eaction$, for short a_b_r) as follows (the long names for arguments is intended to suggest which atoms they are representing):

$a\_b\_r(oxOfAlcohol, hydOfAlcohol, xOfBase, metalOfBase)$

An instance of this reaction is represented below, where the alcohol is ethanol CH_3-CH_2-OH, and the base is sodium hydride Na−H. As can be seen, in this reaction, four atoms change bond, and they are listed as the arguments of the action. In this example, the $metalOfBase$ is the sodium Na on the left hand side of the figure, and the $xOfBase$ is the hydrogen atom attached to the sodium atom.



Introducing action terms is not enough for representing the chemical reactions. We also need to write PAs and SSAs, to specify when the reactions are possible, and what are the effect of them on the fluents (fluents being bonds between the atoms). The PAs are straight forward to write, as they essentially introduce the molecules on the left hand side of the reaction. Consider the PA for a_b_r below:

$$Poss(\, a\_b\_r(oxOfAlcohol, hydOfAlcohol, xOfBase, metalOfBase), s\,) \leftrightarrow$$
$$alcohol(oxOfAlcohol, hydOfAlcohol, s) \wedge base(xOfBase, metalOfBase, s).$$

Notice that the PA gives significance to the arguments of the actions, as it is in the right hand side of the PA that, for example, the first argument of the a_b_r is specified to be the oxygen of the alcohol. Also, notice that the right hand side of this precondition axiom mentions abbreviations $alcohol$ and $base$. The abbreviations defining functional groups and chemical classes can occur only in the precondition axioms and other abbreviations.

The SSAs are slightly more complicated to write, since they should take into account all the bonds that are formed and cleaved as the result of the reaction. For simplicity, assume that we are only interested in the $Bond$ fluent, and that the only action in our knowledge base is a_b_r. Consider the partial formula below as an example of how the effects of a_b_r are captured in the SSA:

$$(\forall x, y, a, s).\ Bond(x, y, do(a, s)) \leftrightarrow$$
$$(\exists\, z, u)(\ a = a\_b\_r(x, z, u, y)) \vee (\ldots)$$
$$\vee\ Bond(x, y, s) \wedge$$
$$(\neg\exists\, u, z)(\ a = a\_b\_r(x, y, u, z)) \vee (\ldots)\,.$$

In the above partial SSA, the first line of the right hand side represents the bonds that will be formed as the result of the reaction. Here, we see that x and y are listed as the first and fourth arguments of the action, which correspond to the oxygen of the alcohol and the metal of the base, respectively. Therefore, if the last action had been a_b_r such that x and y appeared as the first and fourth arguments of the action, then there is a bond between them after the action ($Bond(x, y, do(a, s)$ holds).

The last line of the partial SSA accounts for the bonds that are cleaved as the result of the reaction. Notice that here x and y are listed as first and second arguments of the action, representing oxygen of the alcohol molecule and its hydrogen, respectively. Thus, had the last action been a_b_r such that x and y appear as the first two arguments of the action, then the single bond between them is cleaved after the reaction ($Bond(x, y, do(a, s)$ does not hold).

Finally, the middle line of right hand side of the partial SSA represents the non-effects of the action. It simply states that, if the last action taken neither made the fluent true (i.e. the last action was not mentioned somewhere in the bond formations part in the SSA), nor made it false (i.e. the last action was not mentioned somewhere in the bond cleavages part of the SSA), then the value of the fluent has not changed. In this

case, the last action is ignored and the truth value of the fluent is determined in the previous situation ($Bond(x, y, s)$ determines if x and y have a single bond between them in situation $do(a, s)$).

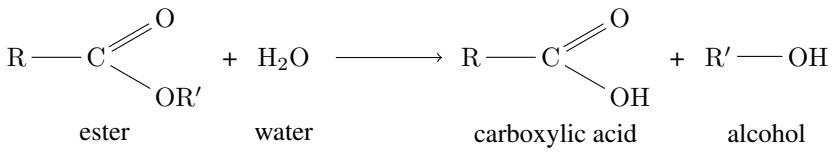
It is worth noting the advantages of our way of representing and reasoning about chemical reactions over representation adopted in the Pathways domain. Most important of all, our approach allows representing generic chemical reactions, while the representation used in the Pathways domain can only support specific instances of reactions. Another important advantage of our approach is that we represent what structural transformations the molecule undergoes. Thereby, our approach provides the means of representing internal mechanism of reactions.

4 Experiments

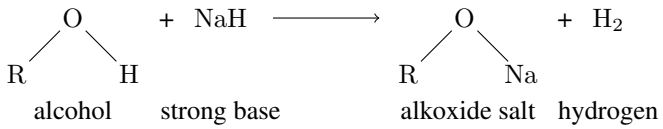
In order to investigate the feasibility of our approach to representing and reasoning about chemical reactions, we implemented it, and tried to solve a number of instances of the organic synthesis problems using several AI planners. More specifically, we encoded instances of the problems in PDDL and PROLOG so that each problem in one language is merely a translation from the other. The experiments were performed on a machine with 2.30 GHz CPU and 12 GB RAM. We ran our tests using a PROLOG planner and two state-of-the-art planners, specifically, Fast-Downward [37] and Roamer [48]. The reason that Fast-Downward and Roamer were chosen in this research was that they support PDDL2.2 [23], which enables representation of realistic planning domains by supporting features like derived predicates and a number of other features. As such, both planners can handle the abbreviations used in our approach. Fast-Downward [37] is a classical planning system based on heuristic search. It is a progression planner, searching the space of world states of a planning task in the forward direction. It uses hierarchical decompositions of planning tasks for computing its heuristic function, called the *causal graph heuristic*, which approximates goal distances by solving a hierarchy of "local" planning problems. The Roamer planner [48] builds on the Fast-Downward planning system, and uses a best-first search in first iteration to find a plan and a weighted A* search to iteratively decreasing weights of plans. Lastly, the PROLOG planner, uses a simple iterative deepening depth-first planning algorithm with declarative heuristics. The declarative heuristics used in the PROLOG planner are domain specific, but not planning problem instance specific. These heuristics identify duplicate unnecessary actions, as well as the irrelevant actions, and consequently reduces the search space. The interested reader can refer to [51] for more information. Preliminary experiments have been also conducted with LPG-td planner (Local search for Planning Graphs [29]). LPG-td is a satisficing planner based on a stochastic local search in the space of particular "action graphs" derived from the planning problem specification and inspired by WALKSAT, an efficient procedure for solving SAT-problems. However, this planner proved unsuccessful for solving any of the instances of the problems we had, as it ran out of memory at the stage of compiling the abbreviations.

The organic synthesis problem instances that were solved required a sequence of four reactions. The generic scheme of these chemical reactions is represented below:

- The reaction between ester and water results in a *carboxylic acid* and alcohol. This reaction needs a strong acid as a catalyzer, but representing the catalyst has been eliminated in the below generic reaction scheme to simplify it.

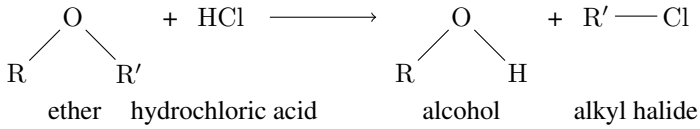


– The reaction between an alcohol and strong base results in an *alkoxide salt* and either water or hydrogen molecule H_2, depending on which strong base is used. Below, the strong base is NaH, and thus H_2 is formed.



– The reaction between alkoxide salt and an *alkyl halide* results in an *ether* and a salt. Here, X is any halogen, and Y is any *alkali metal*. Halogens and alkali metals are specific groups of atoms. For example, Fluorine F and Chlorine Cl are halogens, and Sodium Na is an alkali metal.

– The reaction between ether and mineral acids results in alcohol and alkyl halide. HCl is a mineral acid, and thus the reaction scheme below holds.



Our experiments involved solving 10 different instances of planning problems, each conforming to the above sequence of generic chemical reactions. To clarify the problem instances that were solved, we will present one of them. The other instances are similar to this, and only differ in that they use a different ester molecule in the staring stage. In this problem, our goal is to produce an alcohol molecule and an ethyl chloride $CH_3{-}CH_2{-}Cl$, starting from methyl acetate $CH_3{-}COO{-}CH_3$, water H_2O, hydrochloric acid HCl, sodium hydride NaH and ethyl fluoride $CH_3{-}CH_2{-}F$. Our goal molecules can be synthesized as follows. First, the methyl acetate (an ester), the water molecule, and the hydrochloric acid react to produce methanol (an alcohol) and acetic acid (a carboxylic acid). Notice that hydrochloric acid HCl is the catalyst in this reaction.

$$CH_3COOCH_3 + H_2O + HCl \longrightarrow CH_3COOH + CH_3OH + HCl$$

Second, the methanol (an alcohol) reacts with the sodium hydride (a strong base) to produce sodium methoxide (an alkoxide salt) and H_2.

$$CH_3OH + NaH \longrightarrow NaOCH_3 + H_2$$

Third, the sodium methoxide (an alkoxide salt) reacts with ethyl fluoride (an ethyl halide) to produce methyl ethyl ether (an ether) $CH_2{-}O{-}CH_3$.

$$NaOCH_3 + CH_3CH_2F \longrightarrow CH_3CH_2OCH_3 + NaF$$

Lastly, the methyl ethyl ether (an ether) reacts with the hydrochloric acid (a mineral acid) to produce methanol (an alcohol) and ethyl chloride (an alkyl halide). Notice that our goal molecule of ethyl chloride is only produced at this fourth step, and thus the problem instance required four reactions to achieve the goal.

$$CH_3CH_2OCH_3 + HCl \longrightarrow CH_3OH + CH_3CH_2Cl$$

In our experiments, we represented 9 planning actions (corresponding to 9 different chemical reactions), annotated with 2 relational predicates (used for identifying existing bonds and double bonds between atoms) and 36 derived predicates (for representing the different chemical classes possibly involved in the reactions), in order to make the search space sufficiently challenging. Then, we defined 10 different planning problems of varying complexity by manipulating the composition of the starting molecules in

Chemical Cases	Ester	Prolog Planner	Fast-Downward	Roamer
Case 1	Methyl Acetate	0.06	47.82	47.78
Case 2	Ethyl Acetate	0.12	69.44	69.28
Case 3	Isopropyl acetate	0.75	171.46	171.37
Case 4	Methyl Butyrate	0.13	39.01	38.89
Case 5	Butyl Butyrate	3.45	45.79	45.47
Case 1 + H_2O + HCl	Methyl Acetate	0.06	76.03	75.81
Case 2 + H_2O + HCl	Ethyl Acetate	0.13	111.23	111.01
Case 3 + H_2O + HCl	Isopropyl acetate	0.75	285.56	280.83
Case 4 + H_2O + HCl	Methyl Butyrate	0.13	58.29	58.09
Case 5 + H_2O + HCl	Butyl Butyrate	3.46	67.48	67.43

Table 1. Time performances (in seconds) of the PROLOG planner, Fast-Downward and Roamer.

the planning problem. Specifically, in Table 1, each case refers to a starting stage that includes the following molecules: water H_2O, hydrochloric acid HCl, sodium hydride NaH, ethyl fluoride CH_3-CH_2-F and an *Ester* molecule, that changes from case to case. The experiments done on such starting stages are identifiable in the first 5 rows in Table 1. In order to increase the complexity of the planning problems, we increased the size of the starting stage and compared the effect it has on the performance of the planners. The last 5 rows in Table 1 identify these experiments, where the same chemical cases as above are used, but duplicate water and HCl molecules are available. In all experiments, the goal is to reach a state in which there are atoms identifying an alcohol molecule and ethyl chloride. A planner invoked with whatever of the presented cases generates the aforementioned chain of 4 chemical reactions.

As can be understood from these data, the performance of the PROLOG planner is significantly better than the other planners. This is in part due to the domain-dependent declarative heuristics that are used in the PROLOG planner. In general, constructing domain-dependent heuristics offer opportunities to tailor the mechanisms to the particular domain for far a greater efficiency [47] (typically, several orders of magnitude). A second reason lays in the way the PROLOG planner searches for a path satisfying the goal condition; it reasons on-the-fly on the available knowledge while generating the plan, avoiding to build any intermediate structure to be exploited during the planning task. On the contrary, the state-of-the-art planners we have tested rely on a translator that converts the planner input from PDDL into a multi-valued state representation and on a grounding algorithm used for instantiating operators and axioms (e.g., derived predicates) of the planning domain into a grounded transition system. Then, a search engine exploits the transition system just built for finding a satisfying plan. The main bottleneck is in the grounding algorithm, specifically when a large number of derived predicates need to be instantiated in the transition system, resulting in an exponential blow up of the space required for describing the planning problem.

This is even more apparent when a larger starting stage is experimented (i.e. when duplicate water and HCl exist), where the performance of the PROLOG planner is almost intact while the state-of-the-art planners experience additional performance deficiency. This can be explained by direct relation of size of the starting stage with the time needed for the state-of-the-art planners to build the transition system and employ the domain independent heuristics for finding the solution.

To explore how close are the state-of-the-art planners to their limits, we extended our set of planning instances with an additional reaction: combustion of methanol CH_3-OH. In this reaction, all atoms of two methanol molecules and three O_2 molecules (preconditions of this reaction) change bonds. Therefore, the action representing this reaction needs many arguments (18, to be precise). The initial stage molecules include additional three O_2 molecules and the domain description was extended with two additional abbreviations defining methanol and oxygen molecules. We observed that neither Fast-Downward nor Roamer were able to solve any of the instances of this extended set

of planning instances, as they ran out of memory in their compilation stage. This was somewhat expected because these planners rely on grounding that leads to combinatorial explosion when actions have many arguments. Still, it is somewhat surprising how easy one can push the best planners beyond the edge of their functionality.

The experiments conducted in this paper are relatively simple, and different experimental data may be obtained if more combinatorial benchmark instances (i.e. which require more search) are used. However, we have to emphasize that the contribution of this paper is not only the data obtained from the experiments, rather is also the methodology behind it. We propose an approach that can employ state-of-the-art planners, among others, to solve organic synthesis problems. The experiments that we conducted point out the limitations of the state-of-the-art planners and provide insight about future research directions. At the same time, despite being preliminary, they provide a successful case study of applying AI methods to solve problems in the life sciences.

5 Related Work

There seems to be little earlier work on semantic modeling of generic chemical reactions and organic synthesis, except of our previous work [51]. On the other hand, ongoing research on RuleML and Reaction RuleML includes knowledge representation calculi (e.g., the situation calculus) [7,55,56]. This is one of the emerging Semantic Web technologies that is related to our research. The state of the art in cheminformatics and in ontologies for chemistry is reviewed in [11,12,33,35] where the authors also propose new ontologies for molecules. There are a number of unstructured representations for reactions, e.g., [24]. Similarly to the Pathway Domain [28], internal structure of molecules is not represented and reasoning can be done using only instances of reactions. These approaches have limited scalability because only a small number of reactions can be considered at a time. One could try to build a large graph of specific reactions to reduce the synthesis problem to the reachability problem on this graph, but a graph representation would be infeasible simply because the number of molecules and reactions is very large since there are millions of different alkyls, esters, alcohols and other kinds of molecules. In any case, a graph of specific reactions would not be adequate as a semantic model of generic reactions or as a model of internal structure of participating molecules. Moreover, the industrial cheminformatics systems represent generic reactions [9,42,57] and there is serious ongoing research on developing libraries of generic reactions using both data-driven and model-driven approaches.

The rule based modeling of biochemical systems can be carried out using a set of software tools BioNetGen [6]. However, this approach lacks declarative semantics. BioNetGen is designed mostly for simulating quantitative aspects that require differential equations solvers, or for modeling protein-protein interactions, but not for reasoning about reactions between small molecules. There are a number of representations for biological pathways; Pathway Commons (`http://www.pathwaycommons.org/`) is a collection of pathway data from several publicly available databases (including BioCyc, Reactome and others). These pathways data are represented in BioPAX, the standard exchange format for biological pathways [3,20]. It is defined using the OWL/XML language. The multi-step reactions between small molecules can be described in BioPAX, but there is no representation in ontology for bond changes. The digital files representing molecules in SDF/MOLfile formats [2] can be linked from BioPAX files, but these digital formats include only connection tables between atoms without providing declarative representation for reasoning about bond changes.

Computer-assisted organic synthesis (CAOS), which aims to use computers to help chemists in the process of designing multi-step synthesis of organic compounds, is not a new field. The idea of using a computer-processable language for representing chemical knowledge was proposed in the end of the 1950s, e.g., see [65]. This section is by no

means a comprehensive review of CAOS; for more comprehensive reviews refer to survey papers on this topic, such as [10,15,32,62].

The first synthesis system was organic chemical simulation of synthesis (OCSS) introduced in [16] which is based on retrosynthetic analysis. Retrosynthetic analysis is a technique similar to the well-known notion of regression in SC [58]. Soon after, LHASA [17] was developed which was extension of OCSS by incorporating a more complex strategy system which included multi-step plans based on useful reactions. LHASA uses the knowledge base of reaction transformations and a set of rules to perceive strategic bonds. SECS improves the earlier systems by accounting for stereochemistry (three-dimensional arrangement of atoms in molecules and the effect of this on reactions). OCSS and SECS rely on transform selection by recognition of functional groups and the relationships between them. Two methods of functional group recognition have been described in the literature, one based on decision trees, and the other on matching substructure patterns. Prior to 80s, CAOS systems were based on retrosynthesis approach but in 80s other approaches emerged. SYNGEN [38] was published with the main goal of being able to automatically generate the shortest synthetic route for a given target structure. [27] discussed how machine learning methods can contribute to a self-guided domain-specific heuristic synthesis design system. Later on, the non-interactive system known as SYNSUP was published [21]. So-called "Formal-Logical Approach" is discussed in [63] that focuses on reaction design problems. [68] discusses SYMBEQ computer program created for the search of novel types of organic reactions and is based on this approach. More recently, Route Designer is discussed in [45]. In Route Designer, rules describing retrosynthetic transformations are automatically generated from reaction databases, which ensure that the rules can be easily updated to reflect the latest reactions in the literature. [44] argues how statistical information derived from chemical reaction databases can come handy for predicting the outcome of a reaction, comparable to the most sophisticated methods developed for the same purpose. Lastly, [36] adapts number-proof search to finding a correct synthesis route that can synthesize a goal molecule from commercially available chemical compounds.

6 Discussion and Future Work

In this paper, we employed the expressive and well-studied SC for representing and reasoning about generic (bio)chemical reactions. We are the first to develop semantic modeling for biochemical reactions using an expressive logical language. The approach presented brings forth notable advantages such as being able to represent generic reactions, as opposed to specific instances of reactions. Additionally, it allows reasoning at the level of changing bonds, and this together with the generic reactions adds discovery potential to our approach. Another advantage of our approach is its declarative representation, which makes it easily expandable and flexible. Moreover, our approach is based on SC, which is well-studied in the AI community, and as such any related advancement in the field will be immediate to our approach as well.

One of the current limitations of our approach is that we do not have any negative preconditions for the reactions. Of course, it is not a conceptual limitation of our approach, rather a limitation of the datasets that we considered which lacked the knowledge about the negative preconditions. Additionally, our approach lacks concurrent actions and non-deterministic actions, characterising when some reactions can occur concurrently and when the product of the reactions are not necessarily predictable. Augmenting our reasoning approach with concurent and non-deterministic actions [58] can address this limitation. Another limitation is the lack of representation for stereochemistry, enantiomers, tautomerization and reasoning about chemical reactions in 3D.

As for future work, we intend to scale up the CAOS experiments based on our approach by significantly enlarging the knowledge base of chemical reactions; the library of generic reactions provided by ChemAxon [9] can serve as a potential source, and

try problem instances with more combinatorial search. Additionally, one can explore how our semantic modelling approach can be integrated with existing static ontologies such as those mentioned in the Introduction. Conceptually, it is possible to use our abbreviations to classify molecules in any standard digital format, for example to classify molecules from CHEBI ontology [19,34]. Experimental evaluation of our approach on CHEBI as well as comparison with related research such as [49] remains future work.

The important issue is knowledge acquisition: how are we going to acquire all the axioms that are needed for this approach? We do not expect scientists working in real biological or chemical labs to write anything like PDDL axioms. There are industry-standard digital formats for encoding chemical reactions [2]. Often, these formats can be an output of a graphical tool that a scientist can use to draw molecules and reactions [9], or an output from a data mining tool. We are developing software that can automatically generate our axioms from digital files representing reactions. Our software will use a number of existing open-source cheminformatics software tools.

Acknowledgements. Thanks to anonymous reviewers for comments. The Natural Sciences and Engineering Research Council of Canada and the Dept. of Computer Science of Ryerson Univ. provided partial support for a visit of Andrea Marrella to Toronto.

References

1. Abiteboul, S., Hull, R., Vianu, V.: Foundations of Databases. Addison-Wesley (1995)
2. Accelrys: CTfile Formats (2011), http://download.accelrys.com/freeware/
3. Anwar, N., Bader, G., Demir, E., Donaldson, S., Rodchenkov, I.: BioPAX Biological Pathways Exchange Language. http://www.biopax.org/release/biopax-level3-documentation.pdf
4. Baumann, R., Loebe, F., Herre, H.: Ontology of Time in GFO. In: Donnelly, M., Guizzardi, G. (eds.) FOIS. Frontiers in Artificial Intelligence and Applications, vol. 239, pp. 293–306. IOS Press (2012)
5. BFO: The Basic Formal Ontology. http://www.ifomis.org/bfo (Accessed on 9/20/2013)
6. Blinov, M.L., Yang, J., Faeder, J.R., Hlavacek, W.S.: Graph theory for rule-based modeling of biochemical networks. In: Trans. on Comp. Syst. Biology, LNCS v4230. pp. 89–106 (2006)
7. Boley, H., Paschke, A., Shafiq, O.: RuleML 1.0: The Overarching Specification of Web Rules. In: Proc. 4th Intern. Web Rule Symp. LNCS, vol. 6403, pp. 162–178 (2010)
8. Bry, F., et.al.: Foundations of rule-based query answering. In: Proc. of the 3rd Intern. Summer School on Reasoning Web. pp. 1–153. Springer (2007)
9. ChemAxon: JChem software. http://www.chemaxon.com (accessed on Nov 9 2013)
10. Chen, W.L.: Chemoinformatics: Past, Present, and Future. Journal of Chemical Information and Modeling 46(6), 2230–2255 (2006)
11. Chepelev, L.L., Dumontier, M.: Chemical Entity Semantic Specification: Knowledge representation for efficient semantic cheminformatics. J. Cheminformatics 3, 20 (2011)
12. Chepelev, L.L., Hastings, J., Ennis, M., Steinbeck, C., Dumontier, M.: Self-organizing ontology of biochemically relevant small molecules. BMC Bioinformatics 13, 3 (2012)
13. Chepelev, L.L., Riazanov, A., Kouznetsov, A., Low, H.S., Dumontier, M., Baker, C.J.O.: Prototype Semantic Infrastructure for Automated Small Molecule Classification and Annotation in Lipidomics. BMC Bioinformatics 12, 303 (2011)
14. Claßen, J., Hu, Y., Lakemeyer, G.: A Situation-Calculus Semantics for an Expressive Fragment of PDDL. In: AAAI. pp. 956–961. AAAI Press (2007)
15. Cook, A., Johnson, A.P., Law, J., Mirzazadeh, M., Ravitz, O., Simon, A.: Computer-aided synthesis design: 40 years on. Wiley Interdisciplinary Reviews: Computational Molecular Science 2(1), 79–107 (2012)
16. Corey, E.J., Wipke, W.T.: Computer-assisted design of complex organic syntheses. American Association for the Advancement of Science 166(3902), 178–192 (1969)
17. Corey, E.J., Wipke, W.T., Cramer, R.D., Howe, W.J.: Computer-assisted synthetic analysis. facile man-machine communication of chemical structure by interactive computer graphics. Journal of the American Chemical Society 94(2), 421–430 (1972)
18. Davidson, D.: The logical form of action sentences. In: Rescher, N. (ed.) The Logic of Decision and Action, pp. 81–95. University of Pittsburgh Press (reprinted in "Essays on Actions and Events", 2001, Oxford University Press, pages 105-149, ISBN: 0198246374) (1967)

19. Degtyarenko, K., et.al.: ChEBI: a database and ontology for chemical entities of biological interest. Nucleic Acids Research 36(Database-Issue), 344–350 (2008)
20. Demir, E., et.al.: The BioPAX community standard for pathway data sharing. Nature Biotechnology 28(1-2), 935–942 (2010)
21. Dogane, I., Takabatake, T., Bersohn, M.: Computer-executed synthesis planning, a progress report. Recueil des Travaux Chimiques des Pays-Bas 111(6), 291–296 (1992)
22. Dumontier, M.: Situational modeling: Defining molecular roles in biochemical pathways and reactions. In: Dolbear, C., Ruttenberg, A., Sattler, U. (eds.) OWLED. CEUR Workshop Proceedings, vol. 432. CEUR-WS.org (2008)
23. Edelkamp, S., Hoffmann, J.: PDDL2.2: The Language for the Classical Part of the 4th International Planning Competition. Tech. rep., Albert-Ludwigs-Universitt Freiburg, Institut fr Informatik (2004)
24. Fages, F., Jovanovska, D., Rizk, A., Soliman, S.: BIOCHAM 3.4 Reference Manual. `http://contraintes.inria.fr/biocham/DOC/manual.html` (Oct 2012)
25. Fujita, S.: Description of organic reactions based on imaginary transition structures. Journal of Chemical Information and Computer Sciences 26(4), 205–242 (1986)
26. Gangemi, A., Guarino, N., Masolo, C., Oltramari, A., Schneider, L.: Sweetening ontologies with dolce. In: Gómez-Pérez, A., Benjamins, V.R. (eds.) EKAW. Lecture Notes in Computer Science, vol. 2473, pp. 166–181. Springer (2002)
27. Gelernter, H., Rose, J.R., Chen, C.: Building and refining a knowledge base for synthetic organic chemistry via the methodology of inductive and deductive machine learning. Journal of Chemical Information and Computer Sciences 30(4), 492–504 (1990)
28. Gerevini, A., Dimopoulos, Y., Haslum, P., Saetti, A.: The 5th International Planning Competition: Deterministic part. `http://ipc5.ing.unibs.it/` (2006)
29. Gerevini, A., Saetti, A., Serina, I., Toninelli, P.: LPG-TD: a fully automated planner for PDDL2.2 domains. In: In Proc. of the 14th Int. Conference on Automated Planning and Scheduling (ICAPS-04) International Planning Competition abstracts (2004)
30. Grau, B.C., Horrocks, I., Motik, B., Parsia, B., Patel-Schneider, P.F., Sattler, U.: OWL 2: The next step for OWL. J. Web Sem. 6(4), 309–322 (2008)
31. Gu, Y., Soutchanski, M.: A description logic based situation calculus. Ann. Math. Artif. Intell. 58(1-2), 3–83 (2010)
32. Hanessian, S.: Man, machine and visual imagery in strategic synthesis planning: computer-perceived precursors for drug candidates. Current Opinion in Drug Discovery and Development 8(6), 798–819 (Nov 2005)
33. Hastings, J., Chepelev, L., Willighagen, E., Adams, N., Steinbeck, C., Dumontier, M.: The chemical information ontology. PLoS ONE 6(10), e25513 (2011)
34. Hastings, J., et.al.: The ChEBI reference database and ontology for biologically relevant chemistry: enhancements for 2013. Nucleic Acids Research 41(DB), 456–463 (2013)
35. Hastings, J., Magka, D., Batchelor, C., Duan, L., Stevens, R., Ennis, M., Steinbeck, C.: Structure-based classification and ontology in chemistry. J. of Cheminformatics 4(8) (2012)
36. Heifets, A., Jurisica, I.: Construction of new medicines via game proof search. In: Hoffmann, J., Selman, B. (eds.) AAAI. pp. 1564–1570. AAAI Press (2012)
37. Helmert, M.: The Fast Downward Planning System. J. Artif. Intell. Res.(JAIR) 26, 191–246 (2006), `http://www.fast-downward.org/`
38. Hendrickson, J.B., Grier, D.L., Toczko, A.G.: A logic-based program for synthesis design. Journal of the American Chemical Society 107(18), 5228–5238 (1985)
39. Herre, H.: General formal ontology (gfo): A foundational ontology for conceptual modelling. In: Poli, R., Healy, M., Kameas, A. (eds.) Theory and Applications of Ontology: Computer Applications, pp. 297–345. Springer Netherlands (2010)
40. Hitzler, P., Krötzsch, M., Parsia, B., Patel-Schneider, P., Rudolph, S.: OWL 2 web ontology language. http://www.w3.org/TR/owl2-primer/ (2009)
41. Holliday, G.L., Murray-Rust, P., Rzepa, H.S.: Chemical Markup, XML, and the World Wide Web. 6. CMLReact, an XML Vocabulary for Chemical Reactions. Journal of Chemical Information and Modeling 46(1), 145–157 (2006)
42. James, C., Weininger, D., Delany, J.: Daylight Theory Manual Ver. 4.9 (08/01/11). `http://www.daylight.com/dayhtml/doc/theory/index.html` (2011)
43. Konyk, M., Battista, A.D.L., Dumontier, M.: Chemical knowledge for the semantic web. In: Bairoch, A., Boulakia, S.C., Froidevaux, C. (eds.) DILS. Lecture Notes in Computer Science, vol. 5109, pp. 169–176. Springer (2008)

44. Kowalczyk, B., Bishop, K.J.M., Smoukov, S.K., Grzybowski, B.A.: Synthetic popularity reflects chemical reactivity. Journal of Physical Organic Chemistry 22(9), 897–902 (2009)
45. Law, J., Zsoldos, Z., Simon, A., Reid, D., Liu, Y., Khew, S.Y., Johnson, A.P., Major, S., Wade, R.A., Ando, H.Y.: Route designer: A retrosynthetic analysis tool utilizing automated retrosynthetic rule generation. Journal of Chemical Information and Modeling 49(3), 593–602 (2009)
46. Lloyd, J.W.: Foundations of Logic Programming, 2nd Edition. Springer (1987)
47. Long, D., Fox, M.: Domain-independent planner compilation. In: Working notes of the Workshop on Knowledge Engineering and Acquisition for Planning, held in conjunction with AIPS-98. unpublished material (1998)
48. Lu, Q., Xu, Y., Huang, R., Chen, Y.: The Roamer planner: Random walk assisted best-first search. In: 7th International Planning Competition. pp. 73–76 (2011), http://staff.ustc.edu.cn/~qianglu8/software/seq-sat-roamer.tar.gz
49. Magka, D.: Ontology-based classification of chemicals: a logic programming approach. In: SWAT4LS. CEUR Workshop Proceedings, vol. 952. CEUR-WS.org (2012)
50. Magka, D., Motik, B., Horrocks, I.: Chemical knowledge representation with description graphs and logic programming. In: SWAT4LS. pp. 74–75 (2011)
51. Masoumi, A., Soutchanski, M.: Reasoning about chemical reactions in an expressive logical action theory. In: Discovery Informatics Symposium: 2012 AAAI Fall Symposium Series, AAAI Technical Report FS-12-03. pp. 35–44. AAAI Press (2012), http://www.aaai.org/ocs/index.php/FSS/FSS12/paper/view/5635/5824
52. Mcdermott, D.: The 1998 AI planning systems competition. AI Magazine 21, 35–55 (2000)
53. Motik, B., Grau, B.C., Horrocks, I., Sattler, U.: Representing ontologies using description logics, description graphs, and rules. Artif. Intell. 173(14), 1275–1309 (2009)
54. Özgövde, A., Grüninger, M.: Foundational process relations in bio-ontologies. In: Galton, A., Mizoguchi, R. (eds.) FOIS. Frontiers in Artificial Intelligence and Applications, vol. 209, pp. 243–256. IOS Press (2010)
55. Paschke, A., Boley, H.: Rules Capturing Events and Reactivity. In: Giurca, A., Gasevic, D., Taveter, K. (eds.) Handbook of Research on Emerging Rule-Based Languages and Technologies: Open Solutions and Approaches, pp. 215–252. IGI (May 2009)
56. Paschke, A., Boley, H., Zhao, Z., Teymourian, K., Athan, T.: Reaction RuleML 1.0: Standardized Semantic Reaction Rules. In: Bikakis, A., Giurca, A. (eds.) Rules on the Web, Proc. 6th Intern Symp, RuleML 2012. LNCS, vol. 7438, pp. 100–119. Springer (2012)
57. Pirok, G., Máté, N., Varga, J., Szegezdi, J., Vargyas, M., Dóránt, S., Csizmadia, F.: Making "real" molecules in virtual space. J. of Chem. Inform. and Model. 46(2), 563–568 (2006)
58. Reiter, R.: Knowledge in Action. Logical Foundations for Specifying and Implementing Dynamical Systems. MIT (2001)
59. Smith, B.: Beyond concepts: Ontology as Reality Representation. In: FOIS. pp. 73–84. IOS Press (2004)
60. Smith, B.: Against fantology. Genome Biology 6(1), 153–170 (2005)
61. Thiébaux, S., Hoffmann, J., Nebel, B.: In defense of PDDL axioms. Artificial Intelligence 168(1-2), 38 – 69 (2005)
62. Todd, M.H.: Computer-aided organic synthesis. Chem. Soc. Rev. 34, 247–266 (2005)
63. Tratch, S.S., Zefirov, N.S.: Systematic search for new types of chemical interconversions: Mathematical models and some applications. Journal of Chemical Information and Computer Sciences 38(3), 331–348 (1998)
64. Vladutz, G.: Do we still need a classification of reactions? In: Willett, P. (ed.) Modern Apporaches to Chemical Reaction Searching, Proceedings of a Conference by the Chemical Structure Association of the University of York, England, 8-11 July 1985. pp. 202–220. Gower Publishing Company, Aldershot, England (1986)
65. Vléduts, G.E., Finn, V.: Creating a machine language for organic chemistry. Information Storage and Retrieval 1(2), 101–116 (1963)
66. Vléduts, G.E., Geivandov, E.A.: Automated Information Systems for Chemistry (in Russain). Nauka, Moscow (1974)
67. Yehia, W., Soutchanski, M.: Towards an Expressive Logical Action Theory. In: Proc. of the 25th Intern. Workshop on Description Logics (DL-2012). Rome, Italy (2012)
68. Zefirov, N.S., Baskin, I.I., Palyulin, V.A.: Symbeq program and its application in computer-assisted reaction design. Journal of Chemical Information and Computer Sciences 34(4), 994–999 (1994)