EPiC Series in Computer Science

Volume 36, 2015, Pages 176–194

GCAI 2015. Global Conference on Artificial Intelligence


# Modeling Organic Chemistry and Planning Organic Synthesis

Arman Masoumi[1], Megan Antoniazzi[1], and Mikhail Soutchanski[1]

Dept. of Comp. Science, Ryerson University, 245 Church Street, ENG281, Toronto, ON, M5B 2K3, Canada
mes (at) cs.ryerson.ca

**Abstract**

Organic Synthesis is a computationally challenging practical problem concerned with constructing a target molecule from a set of initially available molecules via chemical reactions. This paper demonstrates how organic synthesis can be formulated as a planning problem in Artificial Intelligence, and how it can be explored using the state-of-the-art domain independent planners. To this end, we develop a methodology to represent chemical molecules and generic reactions in PDDL 2.2, a version of the standardized Planning Domain Definition Language popular in AI. In our model, derived predicates define common functional groups and chemical classes in chemistry, and actions correspond to generic chemical reactions. We develop a set of benchmark problems. Since PDDL is supported as an input language by many modern planners, our benchmark can be subsequently useful for empirical assessment of the performance of various state-of-the-art planners.

## 1 Introduction

Organic chemistry is the chemistry of carbon containing compounds that have important real-life applications. Its research agenda includes several computationally challenging problems. The central problem is Organic Synthesis – the problem of constructing a target molecule from a set of initially available molecules via chemical reactions. Examples include conceiving a sequence of reactions leading to a molecule of penicillin, or to a molecule of chlorophyll (the green pigment in plants). The latter synthesis accomplished by R.B.Woodward in 1960 (51 steps organized in 7 parts) was so striking that it was recognized by a Nobel Prize in Chemistry in 1965. The organic synthesis problem is a highly combinatorial problem since many reactions might be applicable at each step. The idea of using computers as aids in organic synthesis was invented in the end of the 1950s [11; 52]. Subsequent efforts created a new research area known as Computer-Assisted Organic Synthesis (CAOS). The science of organic synthesis was significantly advanced by E.J. Corey who was awarded a Nobel Prize in Chemistry in 1990 "for his development of the theory and methodology of organic synthesis" [11; 12]. In particular, he developed *retrosynthetic analysis*, a problem solving technique somewhat similar to the goal regression in AI [55]. It is known that CAOS is more challenging than chess due to a very high branching factor. The reviews [50], [8] and [9] provide a detailed analysis of CAOS research.

The aim of this paper is to *identify a set of the instances of the organic synthesis problem that can be explored using domain independent planning techniques*. We would like to argue

that organic synthesis is likely to attract interest of the researchers in planning, heuristic search and knowledge representation. Despite earlier understanding that organic synthesis is a kind of planning problem, where reactions serve as actions, and a target molecule is a goal state, there has been no attempt to *formulate organic synthesis in PDDL*, the Planning Domain Definition Language, a popular common language for representing planning problems that is extensively used in the International Planning Competitions [42; 13]. We show how this can be accomplished; this is our first contribution. The model presented in this paper is natural from the perspective of organic chemistry researchers.[1] This is achieved thanks to employing the representation similar to what chemists tend to use when conceptualizing organic chemistry reactions. In this paper, no attempt is made to modify this representation for achieving better computational performance. Our second contribution is in *adopting this model for studying a set of organic synthesis problems using various state-of-the-art planners*. We carry out this study thoroughly, report numerical data and discuss bottlenecks of the current version of the model. Our third contribution is in *raising the questions of how the current model can be modified and what else can be tried to circumvent the existing bottlenecks*, thereby providing enough foundation for the research community to engage and collaborate. The presented model opens several possible directions to be explored in future. Some of the these alternative directions are briefly described in the last section of this paper.

Recently, [25] explored the organic synthesis problem. Inspired by the Corey's retrosynthetic analysis, they formulated the organic synthesis problem as AND/OR graph search and developed a solver based on proof-number search [2]. To evaluate their technique, they have developed a benchmark set of organic synthesis problems based on the publicly available exams given to undergraduate organic chemistry students taking the MIT Course 5.13 "Organic Chemistry 2" [24]. We reworked and elaborated the public chemistry benchmark developed by [24] to make it amenable to domain independent planning. In the work of [25], the set of undergraduate exam problems has been partially solved using an IBM super-computer. Solving it using existing PDDL planners or with other planners on the standard hardware remains a challenge for the AI community. Our goal here is not to propose a solution for this challenge, but rather to *argue that this challenge is likely to stimulate significant new research*. Moreover, we take a step forward to solving this challenge and identify a promising subset of the benchmark that has a chance of being computationally tractable for the state-of-the-art planners. This subset provides the ground for future comparison of alternative approaches, whether obtained from modifying the model proposed in this paper, or those which will use our model as is, but will employ new planning techniques.

A number of interactive computer systems designed to assist a human solving the organic synthesis problem had been developed in the 1960s-1990s. including `LHASA` [10], `SECS` [57], `SYNGEN` [28], `WODCA` [17], `CHIRON` [23], `SYMBEQ` [58]. These systems could work only in a team with an expert chemist who knew how to communicate with the system and who understood the intimate details of the system. There are also a few search-based systems that can run autonomously, e.g., `SYNSUP-MB` [48] does bounded ordered depth-first search, while the system `SYNCHEM` does best-first search in the problem space [37]. A more recent system `ARChem` [38] does automated heuristic guided backward (retrosynthetic) search. Most of the legacy CAOS systems relied on a data base of a few thousand transforms (generalized reactions). Implementations used special purpose procedural representations with chemical knowledge hard-coded into the programs. As far as we know, none of the legacy CAOS systems used domain-independent AI

[1] We presented our modeling approach at a chemistry seminar. Additionally, we discussed it extensively with colleagues who have PhDs in chemistry. All experts in organic chemistry confirmed that our modeling makes sense.

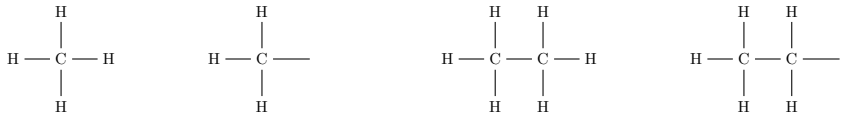
planning.

Despite that the CAOS has a long history, to the best of our knowledge, most of the legacy systems had been developed by the researchers from chemistry [34; 9]. The notable exceptions include the system `SYNCHEM` (`SYNCHEM2`) developed by H. Gelernter and his group [19; 20; 37] that was using backward domain-specific best-first search in the problem space, and the system `SYNLMA`, an expert system for organic synthesis which used a resolution based theorem prover as its reasoning component [51]. To the best of our knowledge, `SYNLMA` is the only system that used a logical encoding for the organic synthesis problem [56]. (We say more about this system in our Discussion section.) Unfortunately, neither the legacy CAOS systems (implemented on legacy hardware and operating systems) nor their benchmarks are publicly available, and therefore, they can not be used for research. Moreover, it was impossible to obtain any instances of the organic synthesis problem that have been previously solved by the earlier CAOS systems. We hope that our PDDL encoding of organic synthesis will serve as motivation to develop more advanced techniques in modeling and planning organic synthesis.

## 2 Preliminaries

This section provides a brief introduction to organic chemistry, and to PDDL, as the modeling language of this application domain.

We consider molecules as graphs, and reactions as symbolic graph transformations. Each molecule is composed of bonded chemical atoms. Chemical atoms can form various types of bonds (single bond, double bond, aromatic bond, etc.) with each other, conditional upon their *chemical valence*, which is a positive integer describing how many bonds a certain atom can make. For example, the valence of carbon atom C is 4, and therefore, a carbon atom is capable of forming 4 single bonds, or 2 double bonds, or a triple bond and a single bond. Hydrogen H has valence of 1, oxygen O has chemical valence of 2, and this is why a molecule of water $H_2O$ consists of 2 hydrogens and single oxygen. Chemical reactivity of molecules is due to their constituent *functional groups*, and molecules with similar properties are categorized into different *chemical classes*. Some of the functional groups are *alkyls*, *hydroxyl* and *ester functional group*, which are the main groups in *alkane*, *alcohol* and *ester* chemical classes, respectively. Alkyl is an acyclic tree of single bonded carbon and hydrogen atoms, with hydrogens in leaves only, such that one of the carbon atoms in the tree bridges it to another functional group via a single bond. If the bridging carbon bonds with a hydrogen atom, an alkane is formed. Alkyls have the generic formula of $C_nH_{2n+1}$ and will be subsequently represented as $R$. If hydrogens in alkane left implicit, it can be considered as a tree of carbons where each node has degree $\leq 4$. The number of alkyls (alkanes) grows fast, e.g., for $n=25$, there are more than $10^7$ different alkanes [43; 16]. Methane is the simplest alkane, methyl is the simplest alkyl. Ethane is the alkane derivative of ethyl, as represented below; ethyl is an alkyl with two carbon atoms as the backbone.



Methane $CH_4$ Methyl $CH_3$ Ethane $CH_3{-}CH_3$ Ethyl $CH_3{-}CH_2$

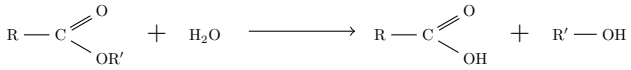
Hydroxyl ($-$OH) is the functional group where an oxygen atom has a single bond with a hydrogen atom, and from the other end forms a single bond with another atom. Alcohols are molecules that contain a carbon atom which has a bond with a hydroxyl, and three other single bonds to some other atoms. Ester chemical class has the generic formula of $R - COO - R'$,

where $R$ and $R'$ are alkyls. Ester functional group is $R - COO-$ with $R$ being an alkyl. The structures of alcohol and ester are displayed below.

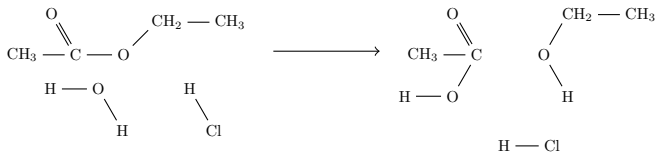
Alcohol Ester

The term *acid* is referred to molecules that donate protons ($H^+$) to other molecules (proton donors). A *base* on the other hand is defined as a proton acceptor. *Hydrochloric acid* HCl is one of the strongest acids.

A chemical reaction is the process in which a set of molecules considered as graphs transform to another set. The molecules existing before a chemical reaction starts are referred to as *reactants* (also known as *substrates* or *reagents*), and the new molecules produced are known as the *products* of the reaction. Chemical reactions are typically modeled as a set of altering bonds, such as Imaginary Transition Structure (ITS) developed by Fujita [15] or "superimposed reaction graphs" developed by G.E. Vléduts [53; 54]. The main idea in these models is that in the course of a chemical reaction, some old bonds cleave, some new bonds form and other bonds do not change. Chemical reactions sometimes need *catalysts* to undergo. Catalysts are chemical compounds that speed up a chemical reaction, but are neither consumed nor transformed to something else. A generic chemical reaction operates with chemical classes and functional groups as opposed to specific molecules, i.e., a generic chemical reaction is a schema that generalizes numerous specific instances of this reaction. For example, consider the generic reaction of hydrolysis of esters, in which an ester molecule reacts with water in presence of a strong acid catalyst which is not displayed, for simplicity; the letters $R$ and $R'$ denote alkyls.



Any molecule that belongs to the ester class can undergo the hydrolysis of esters reaction. For example, ethyl acetate $CH_3{-}COO{-}CH_2{-}CH_3$ (which is an ester with $R$=$CH_3$, $R'$=$CH_2{-}CH_3$) reacts with water $H_2O$ in presence of hydrochloric acid HCl as the catalyst. It is displayed below:



This specific reaction instance complies with the generic hydrolysis of esters reaction schema. Notice that it is infeasible to store all instances of a generic chemical reaction. Moreover, due to the large number of instances of alkyls, alcohols, esters and other functional groups, even for small molecules with at most 10 carbons it would be impractical to store all instances of reactions involving them. Therefore, we need a language for representing generic (re-)actions.

PDDL [42; 18] is the standardized input language for expressing planning problems considered in the International Planning Competition (IPC). Over time, PDDL has expanded to include many features beyond simple classical planning, but its core remains the same. In PDDL, the variables are distinguished by a ? character at front, and dash " − " is used to assign types to the variables and constant objects of the domain. In PDDL, a *domain description* is used to describe the domain of interest, which includes predicates and actions of the domain. It is also

possible to have *derived predicates* [49] that define abbreviations describing those features of the domain which are not affected directly by actions. Derived predicates are commonly used in preconditions of actions, and can be compiled away if they are non-recursive. The specific planning problem instance is represented in a *problem description* file in PDDL, which introduces the objects of the domain in the initial setting and the goal to be achieved.

# 3 Representing Organic Chemistry

In this section we explain how chemical molecules and reactions can be encoded in PDDL. First, we explain how common molecules and functional groups can be represented as derived predicates in PDDL. Then, we show how chemical reactions can be encoded as actions in PDDL.

Molecules are often modeled as undirected graphs, where the vertices of the graph represent the chemical atoms in the molecule, and the edges represent the bonds between the atoms. The same intuition is used here for modeling molecules in PDDL. The atoms are modeled as PDDL objects, and the bonds as relations over the objects, or in PDDL terms, as binary predicates. More specifically, atom types (carbon, oxygen and so on) are determined by PDDL types corresponding to the atom name in the periodic table. For example, the following PDDL code introduces the objects $c1$ and $o1$ as carbon and oxygen atoms respectively.

```
(:objects
  c1 - carbon  o1 - oxygen  x - atom
)
```

Additionally, a type *atom* is introduced which subsumes all other types in the periodic table, such as *carbon*, *oxygen* etc., which is handy when there is no need to be specific.

As for representing the bonds between the atoms, for each type of bond a predicate is introduced. For example, the following introduces the predicates corresponding to single and double bonds, respectively, where $?x$ and $?y$ are variables.

```
(:predicates
      (bond ?x1 - atom ?y1 - atom)
      (doublebond ?x2 - atom ?y2 - atom)
)
```

The other predicate for bonds are defined similarly. The problem description of the PDDL planning problem includes objects and predicates corresponding to atoms and the bonds, which describe the molecules in the initial state, as well as the goal formula. For example, a water molecule in the initial state and in the goal is described as follows:

```
(:objects
   o - oxygen h'- hydrogen h"- hydrogen
)
(:init
    (bond o h') (bond o h")
    (bond h' o) (bond h" o)
)
(:goal
    (and (bond o h') (bond o h"))
)
```

Note that when describing the initial state, each chemical bond is described in both directions, while it is not necessary to do so when describing the goal molecule. This is because each

chemical reaction (which will be discussed shortly) will form and split bonds in both directions. Therefore, if a bond exists in one direction after executing some actions, the bond in other direction will be also present.

Prior to encoding PDDL actions corresponding to chemical reactions, we explain how derived predicates help in formalizing organic chemistry in PDDL. Derived predicates provide a natural way of representing various chemical concepts, including common molecules, functional groups and chemical classes. In addition to convenience, they provide the advantage of modularity since they can be nested. We demonstrate these advantages using examples. The hydroxyl functional group "−OH" can be defined as follows:

```
(:derived
  (hydroxyl ?o - oxygen ?h - hydrogen)
    (and
      (bond ?o ?h)
        (exists (?x - atom)
          (and
            (not (= ?h ?x)) (bond ?o ?x)
          )
        )
    )
)
```

Hydroxyl functional group is used in many common molecules and larger functional groups, including water molecule and alcohol functional group. The latter concepts can be represented as derived predicates as well, which use hydroxyl in their definition:

```
(:derived
  (water ?h - hydrogen ?o - oxygen)
    (and
      (bond ?o ?h)
        (exists (?h2 - hydrogen)
          (and
            (not (= ?h ?h2))
            (hydroxyl ?o ?h2)
          )
        )
    )
)

(:derived
  (alcohol ?c - carbon ?o - oxygen)
    (and
      (bond ?o ?c)
      (exists
        (?h - hydrogen
         ?x1- atom ?x2 - atom ?x3 - atom
        )
          (and
            (hydroxyl ?o ?h)
            (not (= ?x1 ?x2))
            (not (= ?x1 ?x3))
```

```
            (not (= ?x1 ?h))
            (not (= ?x2 ?x3))
            (not (= ?x2 ?h))
            (not (= ?x3 ?h))
            (bond ?c ?x1)
            (bond ?c ?x2)
            (bond ?c ?x3)
          )
      )
    )
)
```

Although in our approach the arguments of the derived predicates account for only a few nodes in the molecule graph they are representing, they identify the whole molecule and are referred to as the *key atoms* of the molecule. For example, in the above definitions for water and alcohol, the atoms $o$ and $h$ identify the molecules as a whole. The key atoms are often chosen from the atoms at the common reaction sites.

In general, one needs recursive derived predicates, for example when defining arbitrary alkyls. Encoding an arbitrarily branching alkyl requires the transitive closure over edges of the tree representing the molecule to exclude cycles, and defining transitive closure requires recursion. However, definitions of small alkyls can be unfolded into non-recursive abbreviations. In our encoding, derived predicates are used in preconditions of actions as explained below.

Each generic chemical reaction is represented using a PDDL action, whose effects cleave and form all the respective bonds that the generic reaction alters. For example, consider the action representing hydrolysis of esters reaction:

```
(:action hydrolysis_of_esters
  :parameters
    (?c1 - carbon ?o1 - oxygen
     ?h1 - hydrogen ?o2 - oxygen
     ?h2 - hydrogen ?x - atom)
  :precondition (and
    (ester ?c1 ?o1)
    (water ?h1 ?o2)
    (strong_acid ?h2 ?x))
  :effect (and
    (not (bond ?c1 ?o1))
    (not (bond ?o1 ?c1))
    (not (bond ?h1 ?o2))
    (not (bond ?o2 ?h1))
    (not (bond ?h2 ?x))
    (not (bond ?x ?h2))
    (bond ?c1 ?o2)
    (bond ?o2 ?c1)
    (bond ?o1 ?h2)
    (bond ?h2 ?o1)
    (bond ?h1 ?x)
    (bond ?x ?h1))
)
```

Notice that derived predicates are used in the preconditions of the action to characterize

which molecules are needed for the reaction to be applicable. Moreover, all the atoms that change bonds during the chemical reaction must be listed as parameters of the action, otherwise we could not specify all effects of the generic reaction. When specifying the effects, we make sure that the action forms and splits the bonds in both directions. In the hydrolysis of esters chemical reaction, six atoms change bonds. Specifically, the key atoms of an ester molecule, a water molecule and a strong acid split their bonds (all of them have been indicated by arrows on the left hand side of Figure 1), and new bonds form as displayed by arrows on the right hand side of Figure 1. For example, the carbon atom and the oxygen atom in the ester molecule have a bond before the reaction, which splits after the reaction. Conversely, there is no bond between the carbon atom of the ester molecule and the oxygen atom of the water molecule before the reaction, but it forms after.

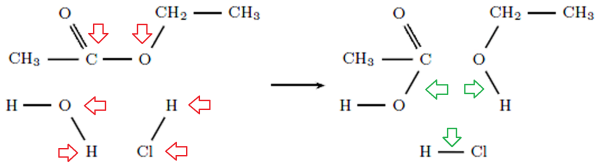


Figure 1: Hydrolysis of ethyl acetate

## 4 Experiments

To create a set of instances of the organic synthesis problem amenable to the state-of-the-art planners, we reworked a set of benchmark problems publicly available from [24]. This set of benchmark problems includes 20 organic synthesis problems (called the set20 below, for brevity) collected from MIT undergraduate organic chemistry exams. The set20 includes a library of 62 starting molecules, and 50 definitions of reactions encoded using a line notation popular in chem-informatics [33]. The easiest problem can be solved using 3 reactions, the longest requires 12 reactions, while the majority of the problems can be solved using 4-5 reactions. As reported in [25], their program running on an IBM super-computer solved 15 out of the 20 problems within 6 hours cut-off time by using the proof-number search. This technique involves backward search from a target molecule and works well with incompletely specified reactions on the input. In particular, some effects of the reactions can be left unspecified, and the reactions in the knowledge base can be unbalanced, e.g., information about the products can be missing. Also, their program delegated all chemistry related reasoning, e.g., deciding sub-graph isomorphism, to specialized proprietary software [7]. The knowledge base in [24] was completely adequate for solving the problems in the set20 using their technique, but it was not directly usable for our purposes. In PDDL, all effects of the actions must be explicitly specified. Therefore, we added manually the missing knowledge to make all the reactions as chemically complete as possible. This was laborious and time consuming process that required consultations with chemistry experts. Since some of the reactions in [24] combine two reactions, we separated them and obtained as a result a knowledge base of 52 reactions. For a few reactions, our preconditions are weaker than what is needed in reality for the reaction to occur, i.e., any reaction that happens in reality is deemed possible in our approach, but not necessarily the other way around. This is done to closely match representation of the reactions from the set20. Once all processing has been done, we ran an in-house developed Python program to translate all the completed generic reactions from the RXN format popular in industry [1] into a PDDL representation described in the previous section. Similarly, we translated the planning problems

into our PDDL representation. It is noteworthy that the derived predicates in the preconditions of the reactions in the set20 were all representable without the need for recursion since they mentioned only small alkyl molecules with at most 4 carbons. To represent these alkyls we manually encoded each alkyl containing from 1 up to 4 carbon atoms as a derived predicate, and then defined a general alkyl derived predicate in terms of the disjunction of the predefined specific alkyls. Having no recursive derived predicates facilitates compiling away the derived predicates from the preconditions of actions.

All the experiments were performed on a machine with 2.80 *GHz* CPU and 128 *GB* of RAM. We experimented with a number of planners, some of which support PDDL 2.2 [13] while the others do not. We are able to use planners that do not support PDDL 2.2 since we can compile away the derived predicates from the preconditions of the reactions, and thereby produce a domain with no derived predicates. The planners supporting PDDL 2.2 that were used for experimentation were FastDownward (downloaded on July 31, 2014), abbreviated FD, [26], LPG-td version 1.2 [22], ROAMER-p, abbreviated RMR, [39] and MIPS-XXL, version 3, released 29th of June 2007 [14; 32]. FastForward, version 2.3, [29] (abbreviated FF) was the only planner not supporting PDDL 2.2.

Not surprisingly, none of the translated planning problems from the set20 could be solved using any of the aforementioned planners, as they all exceed the available memory before finding a solution. The root of the problem appears to be in the large number of variables in the actions (and derived predicates if they were included) which results in a combinatorial explosion when grounding the domain description into a propositional representation. To clarify, note that there are 52 reactions, number of arguments of which range from 4 to 13. Depending on the planning problem in the set20, the number of atoms (objects) varies, but typically there are more than 30 atoms in a problem, usually 10 or more for each of carbon, hydrogen and oxygen types. Additionally, in the case when derived predicates are included, there are 36 derived predicates, each of which can have 15 or more local variables to instantiate. For example, consider a (re-)action with 10 arguments that for the purposes of grounding has to be instantiated, and let us assume four arguments are carbon atoms, three are oxygen and remaining three are hydrogen atoms. Assuming 10 atoms for each type, this results in $10^4 * 10^3 * 10^3 = 10^{10}$ possible instantiations, just for this action. Even if many of these instantiations can be ruled out a priori by the existing grounding algorithm, the remaining number is too large for the planners dealing with the current PDDL encoding of the problem.

To identify a more promising benchmark we simplified the problems in the set20. The simplification strategy for the problems had two dimensions: (i) we removed a few reactions to reduce the size of the domain, and (ii) we simplified the goal of the planning problem and reduced the number of atoms in the initial state. Specifically, instead of the goal molecules in the set20, we consider as new goals intermediate molecules produced by the first and second reactions solving the synthesis problems in the set20. For book-keeping purposes, we preserve same enumeration of the planning problems as in the set20. For example, the organic synthesis problem 20 would give rise to simpler planning instances that can be solved by 1 reaction, or by 2 reactions. After these simplifications of the goals, we obtained a new set of planning instances, where the target molecule can be synthesized in 1, 2 or 3 steps. All the extra atoms no longer needed for synthesis of a simpler goal molecule were removed from the initial state. Furthermore, we explored two alternatives. In one case, we compiled away the derived predicates, but kept the useless derived predicates listed in the domain description, although they were never referenced by any action or goal in the problem. In another case, once the derived predicates have been compiled away, we removed all of them from the domain description. In the first case, we could experiment only with the planners supporting PDDL 2.2, while in the second case, we could

work with FF too.

At the initial stage, we reduced the number of reactions to 36. The number of arguments in actions varied from 4 to 13. If we kept the unused derived predicates, the domains also included 36 derived predicates. In total, 26 simpler planning problems were created, which required plans of length 1-3 in order to be solved. Out of 26, 15 planning problems required a solution of length 1, 6 problems – a solution of length 2, and 5 problems – solutions of length 3.

In the case where all derived predicates were removed from the domain, regardless of the choice of the planners, no problem that required a plan of length 3 was solved, and only one problem that required a plan of length 2 were solved. In all the unsuccessful cases, the memory of the machine was exhausted prior to finding any plan. Table 1a presents the time in seconds it took the planners to solve the problems, averaged over 5 runs, only for the successful cases, where a solution was found. MIPS planner was unsuccessful in solving any of the problems, so it does not appear in the table. The problem numbers correspond to their counterparts in set20. We verified manually that all found reactions are actually correct solutions. The dashes in the table mean that the specific planner was unable to solve the specific problem, again due to memory exhaustion. The standard deviations for all the planners and problems across the runs are insignificant, except for the problem 20 of length 2, which the standard deviation for LPG-td was 658 seconds. It is noteworthy that for FD which works in three stages, the times for the stages are added up in table 1. Table 1b presents the corresponding memory usage on the problems in MBs, where for FD the maximum usage across the stages is considered.

| # | L | FD | FF | LPG | RMR |
|---|---|---|---|---|---|
| 3 | 1 | 4.24 | 0.05 | 5.24 | 4.12 |
| 6 | 1 | 1.32 | 0.01 | 0.24 | 1.08 |
| 7 | 1 | 346.62 | 0.91 | 4.07 | 334.41 |
| 8 | 1 | 348.23 | — | — | — |
| 12 | 1 | 938.95 | 30.30 | 80.52 | — |
| 14 | 1 | 4.07 | 0.05 | 5.26 | 3.92 |
| 20 | 1 | 50.34 | 1.32 | 125.29 | 212.55 |
| 20 | 2 | 117.53 | 2.05 | 730.73 | 319.87 |

(a) Time (sec) performances with 36 actions.

| # | L | FD | FF | LPG | RMR |
|---|---|---|---|---|---|
| 3 | 1 | 108 | 48 | 176 | 86 |
| 6 | 1 | 50 | 13 | 139 | 28 |
| 7 | 1 | 2375 | 790 | 693 | 2702 |
| 8 | 1 | 12631 | — | — | — |
| 12 | 1 | 12375 | 15566 | 18043 | — |
| 14 | 1 | 108 | 48 | 176 | 86 |
| 20 | 1 | 1067 | 870 | 1143 | 7361 |
| 20 | 2 | 2039 | 1225 | 1635 | 9333 |

(b) Memory (MBs) usage with 36 actions.

Table 1: The domain with 36 actions.

Interestingly enough, in the case where the useless derived predicates were not removed from the domain, no problem was solved, regardless of the choice of the planner. In every instance, the process was killed before finding a solution as the planner exhausted all available memory. Since presence of useless derived predicates is the only difference between this case and the previous case, and since some of the problems in the previous case were solved while all problems failed in this case, we can conclude that the addition of useless derived predicates considerably worsens the performance of the planners.

Hoping to find an even more promising sub-set of the benchmark set20, we simplified the problems even more, by removing actions from the domain with greater number of arguments. Since presence of useless derived predicates is an overhead for the planners, we removed all of them from the domain description. We ruled out MIPS due to its poor performance. The new domain contained 23 actions with 4-6 arguments. As a result of excluding 13 actions from the domain, some problems from the previous stage were invalidated since their solutions rely on the actions we eliminated. A subset of 9 problems was solvable using this new domain, with 8

of them requiring a single reaction, 1 of them – a sequence of two reactions. In this stage, FD, FF and LPG-td were able to solve 1 more organic synthesis problem than they could with the larger domain. ROAMER was not able to solve any additional problems. Table 2a displays the times in seconds corresponding to this attempt, where only successful results are presented. The problems that were not solved exhausted the available memory. All the descriptions for Table 1 applies to Table 2 as well. As for the standard deviation across 5 runs, the only significant values are for problem 10 and FD, which has 2587 seconds standard deviation, problem 20 (length 2) and LPG-td with 657 seconds. Table 2b presents the memory usage in MBs.

| # | L | FD | FF | LPG | RMR |
|---|---|---|---|---|---|
| 3 | 1 | 3.67 | 0.05 | 5.10 | 3.52 |
| 6 | 1 | 0.97 | 0.01 | 0.24 | 0.90 |
| 7 | 1 | 2.51 | 0.07 | 2.31 | 2.19 |
| 8 | 1 | 133.3 | 335.97 | 960.83 | — |
| 10 | 1 | 8947.6 | — | — | — |
| 12 | 1 | 930.6 | 29.46 | 390.91 | — |
| 14 | 1 | 3.80 | 0.05 | 5.13 | 3.48 |
| 20 | 1 | 21.33 | 1.20 | 90.17 | 174.9 |
| 20 | 2 | 69.87 | 1.79 | 729.87 | 255.6 |

(a) Time (sec) performance with 23 actions.

| # | L | FD | FF | LPG | RMR |
|---|---|---|---|---|---|
| 3 | 1 | 101 | 47 | 175 | 78 |
| 6 | 1 | 45 | 13 | 138 | 24 |
| 7 | 1 | 76 | 64 | 192 | 54 |
| 8 | 1 | 1571 | 92527 | 105447 | — |
| 10 | 1 | 30175 | — | — | — |
| 12 | 1 | 12695 | 15539 | 18003 | — |
| 14 | 1 | 101 | 47 | 175 | 78 |
| 20 | 1 | 384 | 864 | 1116 | 6466 |
| 20 | 2 | 1319 | 1215 | 1613 | 7888 |

(b) Memory (MBs) usage with 23 actions.

Table 2: The domain with 23 actions.

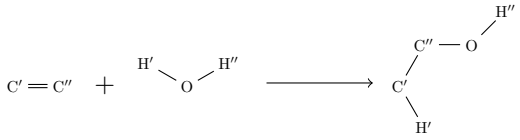
In all the cases where the planners failed to find a solution, the planners fail during the grounding process. The huge numbers of grounded actions and fluents had been generated that exhausted all available memory. Also, at least 95% of the total time in the reported tables is taken by grounding and preprocessing, i.e., finding an action is fast. As for comparison of the planners, FF works best in terms of speed of finding the solutions. However, FD works best in terms of being successful in solving the most organic synthesis problems in the benchmark that we designed. The superiority of FF in terms of speed over other planners is partly due to short planning problems in the benchmark, and partly due to FF's strategy to perform incomplete greedy search EHC before switching to complete BFS search. In terms of memory requirements, there is no clear distinction between the planners, but overall FF needs less memory than its competing planners.

# 5 Examples

To illustrate our experiments and results, we provide a specific example of the organic syntheses problem that the AI planners tried to solve. Our example is related to the first and second reactions solving the problem 20, i.e., it is related to the one-step and two-step versions of this problem included in the tables above. We start with explaining the reactions that solve the problems, then we show initial and goal molecules. Both reactions are part of the set of 23 actions mentioned above.

The first generic reaction happens between water and an alkene, a molecule composed from carbons and hydrogens only that has the single double bond between two carbon atoms. This reaction is usually catalyzed with a diluted acid, but we leave it out for simplicity. In chemistry, this reaction is known as acid-catalyzed hydration of alkenes; it belongs to the class

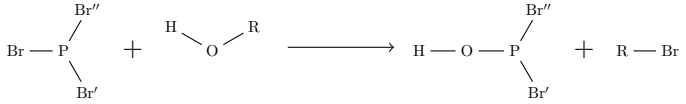
of addition reactions, e.g., see Section 6.9 in [4]. In the figure illustrating this reaction, we assume that the distinct carbon atoms $C', C''$ stand for single-bonded chains of carbon atoms, which may possibly form a ring, such that all remaining valences of the carbons are saturated with hydrogens. Since shapes and the number of carbons in these chains can vary, the figure illustrates a generic reaction.




The reaction breaks the bond between the oxygen atom $O$ and the hydrogen atom $H'$, and forms the bond between the carbon atom $C'$ and $H'$, while the double bond between $C'$ and $C''$ converts into a single bond, and the carbon atom $C''$ bonds with $O$. If we denote carbon atoms $C', C''$ with the variables `?c_4,?c_5`, respectively, and hydrogen atoms $H', H''$ with the variables `?h_2,?h_3`, respectively, then this generic reaction can be rendered in PDDL as follows.

```
(:action alkeneAndWater
  :parameters
     (?o_1 - oxygen ?h_2 - hydrogen ?c_4 - carbon ?c_5 - carbon)
  :precondition
     (and (not (= ?c_5 ?c_4))
          (exists (?h_3 - hydrogen)
             (and (not (= ?h_2 ?h_3)) (bond ?h_2 ?o_1) (bond ?h_3 ?o_1) ) )
          (doublebond ?c_5 ?c_4)
     )
  :effect
     (and (not (bond ?h_2 ?o_1)) (not (bond ?o_1 ?h_2))
          (bond ?o_1 ?c_5) (bond ?c_5 ?o_1)
          (bond ?h_2 ?c_4) (bond ?c_4 ?h_2)
          (not (doublebond ?c_5 ?c_4)) (not (doublebond ?c_4 ?c_5))
          (bond ?c_5 ?c_4) (bond ?c_4 ?c_5)
     )
)
```

The second generic reaction uses phosphorus tribromide $PBr_3$ that reacts with alcohols R−OH to produce alkyl bromides R−Br, where R stands for an alkyl or a cyclo-alkyl (a ring of carbon atoms). This is one of the methods of preparing alkyl bromide, e.g., see Section 4.14 in [4]. In figure below, we use notation $Br$, $Br'$ and $Br''$ to show that all three bromine atoms are distinct, all of them are bonded with the phosphorus atom P before the reaction, but one of them splits from P and bonds with R. This reaction also splits the bond between the oxygen atom O and R, and forms the bond between O and P. The bond between the hydrogen atom H and O remains intact.




Since we have to unfold all derived predicates such as alkyl, alcohol and hydroxyl, we represent this reaction in PDDL using the following action schema with weakened precondition and abbreviations compiled in. The variable `?c_7` can match any carbon with the bond to hydroxyl, and therefore, it can also match a key carbon atom (a reaction site) designating an alkyl or cyclo-alkyl $R$. A false-positive match can happen in general due to weaker precondition, but it

does not occur in our test cases.

```
(:action alcoholAndPBr3
  :parameters
     (?o_5 - oxygen ?br_4 - bromine ?c_7 - carbon ?p_1 - phosphorus)
  :precondition
     (and (exists (?br_3 - bromine ?br_2 - bromine)
         (and (not (= ?br_4 ?br_3)) (not(= ?br_2 ?br_4)) (not(= ?br_2 ?br_3))
         (bond ?br_2 ?p_1) (bond ?p_1 ?br_4) (bond ?p_1 ?br_3)))
         (exists (?h_6 - hydrogen) (and (bond ?h_6 ?o_5) (bond ?o_5 ?c_7)) )
     )
  :effect
     (and (not (bond ?p_1 ?br_4)) (not (bond ?br_4 ?p_1))
          (bond ?p_1 ?o_5) (bond ?o_5 ?p_1)
          (bond ?br_4 ?c_7) (bond ?c_7 ?br_4)
          (not (bond ?o_5 ?c_7)) (not (bond ?c_7 ?o_5))
    )
)
```

In action schemas, subscripts correspond to atom numbers, e.g. `?p_1` represents a phosphorus atom P that is assigned the number 1, `?br_4` represents the bromine atom Br that is assigned the number 4, and `?br_2,?br_3` represent the bromine atoms $Br', Br''$ assigned the numbers 2 and 3, respectively. Each atom number identifies uniquely an atom before the reaction and after the reaction that must be complete and balanced. The atom numbers simplify the task of tracing the effects of the reaction.

The initial molecules for the 1-step version of the problem 20 are water and a cyclohexene molecule $C_6H_{10}$ that is an instance of an alkene since it has only one double bond. In Figure 2 displaying this molecule, we follow convention common in chemistry that hydrogen atoms remain implicit. Since valence of carbon is 4, and valence of hydrogen is 1, implicit hydrogen atoms can be easily restored.

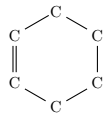


Figure 2: Cyclohexene molecule $C_6H_{10}$ is an alkene. Hydrogen atoms are not shown.

The initial molecules for a 2-step version of the problem 20 includes not only water and cyclohexene, but also phosphorus tribromide, see above. The goal molecules are the following.

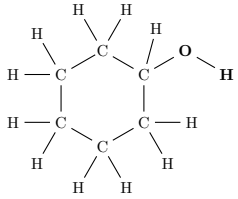


Figure 3: Goal in a 1-step problem

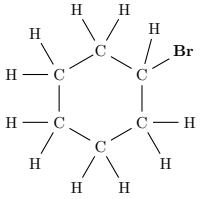


Figure 4: Goal in a 2-step problem

Notice that in comparison to the initial cyclohexene molecule the goal molecule in Figure 3 has a hydroxyl group $-$OH attached, while in the goal molecule in Figure 4 hydroxyl is replaced by

a bromine atom Br. The only double bond in cyclohexene is replaced by the single bond when the alkene and water reaction is executed. In Figures 3 and 4 we display all hydrogen atoms.

In the 1-step version of the problem 20, we have 6 carbon atoms, 12 hydrogen atoms (10 out of which belong to cyclohexene, and 2 constitute a water molecule), and one oxygen atom. In total, there are 19 atoms which is a relatively small number. Therefore, the grounded instances of the 23 actions available in the domain description can fit in the computer RAM. This is why the planners can find successfully that the alkene and water reaction is the correct solution out of 23 given actions in the domain description. In the 2-step version of the problem 20, we have additionally 3 bromine atoms and 1 phosphorus atom, but the total number of atoms is still manageable for the AI planners. For example, consider the alkene and water reaction $alkeneAndWater(?o_1, ?h_2, ?c_4, ?c_5)$. A planner would consider a subset of $12 \cdot 6 \cdot 6$=432 instantiations of this action, and many ground instances would be rejected using preconditions. Similarly, when instantiating the $alcoholAndPBr3(?o_5, ?br_4, ?c_7, ?p_1)$ action, only a subset of $3 \cdot 6$ groundings would be considered, but for each of them $3 \cdot 3 \cdot 12$ instances are evaluated due to existential quantifiers in preconditions. Apparently, for all the remaining 21 action schemas the total number of instantiations is similarly small. Therefore, the planners find that the alkene and water reaction followed by the reaction between an alcohol and phosphorus tribromide is the sequence of actions that produces a goal molecule. However, the remaining steps in solution to the benchmark problem 20 require more initial molecules, and consequently, the larger number of initially given atoms; a solution to the problem 20 from the benchmark needs more than 10 molecules with 58 atoms in total. As a consequence, the total number of ground instances would exceed allocated memory. One possible approach to dealing with the grounding problem would be to develop a domain specific grounding algorithm that would prune more intelligently incorrect instantiations using chemistry specific knowledge. However, this would defeat the challenge of solving the problem using domain independent techniques.

## 6 Discussion

It is worth noting two main differences of our way of representing reactions from a related approach used in the *Pathways domain* [21]. In Pathway domain, a reaction is modeled as a mechanism transforming a set of specific molecules (each of which is represented as an object constant) to a different set of specific molecules. This approach is only capable of representing specific reactions, while our methodology supports representation of generic chemical reactions as well, which is an important advantage due to a huge number of known reaction instances. Moreover, as a pragmatic argument in favor of representing generic reactions, one can mention that the digital RXN format [1] currently popular in chem-informatics industry is designed to represent generic reactions.

We anticipate that knowledge acquisition will not be bottleneck in constructing large knowledge bases of generic reactions in PDDL. There are publicly accessible repositories of generic chemical reactions represented in industry-standard formats [1]. To acquire the large knowledge base of actions representing generic reactions, it is possible to automatically translate these repositories into our representation in PDDL. The current model has a limitation though, since stereo-chemistry is not represented. New fluents can be introduced to represent 3D features of the organic molecules. This remains for future work.

Regarding related research, [24; 25] did not attempt representing the organic synthesis problem in PDDL. Their implementation was relying on chem-informatics software developed by ChemAxon [7]. Moreover, there is nothing in their work that can help with developing a PDDL representation. Their reactions are incompletely specified using a specialized chem-informatics

language, and for this reason they cannot be studied using domain independent AI planning techniques, since the latter require that all effects and all preconditions of the reactions must be explicitly represented. In our previous paper [40], we developed a situation calculus based representation for generic reactions. However, we did not develop a PDDL representation, and did not attempt to solve the synthesis problems using the state-of-the art planners. Lastly, there are three main differences between our work and the paper [41]. Firstly, in this paper, we identified a realistic benchmark for further research. Secondly, our work conducts a rich experimental study by considering a larger realistic domain (in comparison to the 4-action domain used in [40; 41]) and much elaborated analysis of the results. Thirdly, as opposed to manual encoding employed in our previous papers, this work involved developing software that automatically translates generic reactions from the industry standard RXN format into PDDL actions. Please note this also *de facto* validates accuracy of our modeling.

To the best of our knowledge, the system `SYNLMA` [51; 56] is the only other CAOS system using logic-based knowledge representation techniques. We learned about this legacy system recently, when our research has been already completed. It turns out that our representation is very different from `SYNLMA` where molecules are divided into fragments that are represented using the predicate $Fragment(x)$ whose argument is a term describing several bonded atoms in the given molecule. There are many function symbols that can be used to describe different kinds of the fragments. Therefore, each molecule may have many different representations in comparison to our approach where each molecule has an unique representation as the conjunction of statements about bonds connecting atoms. In `SYNLMA`, reaction rules are represented as pairs of clauses built using the predicate $RxnRule$. The arguments of this predicate are linked lists used to trace the substitutions performed during unification. It is clear that representation from `SYNLMA` cannot be directly translated to PDDL. In `SYNLMA`, the organic synthesis problem is reduced to a sequence of automated reasoning problems. The system used a layered approach where theorem proving on the bottom layer is controlled by a heuristic AND/OR graph search process on the top layer. According to empirical assessments reported and discussed in [56], `SYNLMA` attempted to produce one reaction step for synthesizing the molecule propoxyphene (used in a safety-plagued pain-killer drug Darvon from the 1950s) with 25 non-hydrogen atoms, but this synthesis required interaction with an user. Also, [56] reports other experiments carried out with simpler molecules containing only 5 to 10 non-hydrogen atoms, but efforts focused on finding sub-molecules rather than on solving the organic synthesis problem.

Unfortunately, none of the legacy and commercial CAOS systems are publicly available. Therefore, we cannot evaluate their performance using our benchmark.

# 7 Future Work and Conclusion

As expected and shown by our experiments, the complete version of the reworked benchmark from [25] modelled as discussed in this paper is too computationally challenging for the state-of-the-art planners which work by grounding the problem. There appears to be two main bottlenecks causing the computational difficulties. The first is the derived predicates. Although useless in these experiments, they create a vast amount of overhead for the planners. The second is actions with a larger number of arguments to be instantiated. In combination with typically large numbers of objects necessary for organic synthesis problems, these bottlenecks result in a combinatorial explosion. Our simplification strategies and experimental studies helped us identify two sizable subsets of the benchmark (the sets with 36 actions and 23 actions) such that a number of the planning instances in these subsets can be solved using the state-of-the-art planners. These subsets provide a foundation for comparison of the alternative approaches that

can be developed as future work.

There are at least four different directions for future work that can take advantage of the subset of the benchmark that we identified. One would be reformulating the current model aiming to mitigate the problems that occur as a result of grounding. Our current model leaves room for many alternative approaches in this direction to be explored in future. For example, although on the surface the model is using ADL constructs, it can be transformed to comply with the STRIPS requirements. As discussed in the Experiments section, thanks to finite stratification of the derived predicates, the derived predicates in our model can be compiled away in action preconditions. Also, the PDDL *exists* blocks can be eliminated by introducing extra parameters in the actions. Transforming the model to STRIPS enables using automatic domain modification techniques such as action schema splitting [3]. Another approach could be introducing dummy actions ( for example *select_atom* with one argument of type atom) and additional predicates stating which atoms were selected. These auxiliary actions help to transform actions representing chemical reactions into actions without arguments, and therefore alleviate the grounding problem mentioned earlier. However, it is noteworthy that application of such domain modification techniques does not guarantee success, since they increase the length of the plan. These domain modification techniques might be promising for short problems, but they might be weak for realistic challenging organic synthesis problems which are typically long. Another disadvantage of approaches based on reformulating the current model is that the modifications will come at the cost of reducing the naturalness of the model for this domain.

The second possible direction of research would be keeping the current model as is, but considering lifted planning that postpones or minimizes grounding. A relevant work in this direction would be [45; 44]. However, similar to the previously discussed direction, the real challenge is not in solving simple 1-2 step problems, but in solving the instances from the set20 and, if successful, then solving longer organic synthesis problems.

The third direction might be related to developing a more efficient general grounding algorithm that instantiates PDDL action schemas. The currently popular grounding algorithms and related computational trade-offs are discussed in Section 6 of [27]. Developing an efficient domain specific grounding algorithm that prunes away redundant instantiations using chemistry knowledge might be a promising approach, if one would like to trade generality for efficiency.

Finally, it would be interesting to explore encodings of planning that allow to reduce planning to solving satisfiability of Quantified Boolean Formulae (QBF). In this respect, recently developed partially grounded QBF encoding would be particularly promising [6; 5]. Since state-of-the-art SAT solvers are much more mature than QBF solvers, and reductions of planning to satisfiability are well known [36; 35; 47; 46; 30; 31], exploring compact and lifted SAT encodings of the organic synthesis problem would be also an interesting direction in future.

As a conclusion, we propose the organic synthesis problem as a new invaluable application domain for planning that may help develop more advanced techniques in modeling and planning. Since PDDL is a generally accepted input language supported by many modern planning systems, we hope that our PDDL benchmark will help to stimulate new research on planning.

## Acknowledgements

Thanks to the Natural Sciences and Engineering Research Council of Canada (NSERC) for long-term funding support to Mikhail Soutchanski within NSERC's Discovery Grant program. This grant provided financial support to Arman Masoumi for his Master's thesis research in 2012-2014. Megan Antoniazzi's work was financially supported in summer 2014 by an undergraduate research assistantship funded by the Department of Computer Science at Ryerson University.

The Department of Computer Science has provided access to the private cloud, which enabled us to run our experiments on a Linux-based virtual machine that has been allocated 128GB of RAM. Vitaliy Batusov contributed a Python program converting reactions from RXN to PDDL. Thanks to Malte Helmert for discussions of Fast-Downward and PDDL. Thanks to anonymous reviewers of a preliminary version of this paper for constructive feedback.

## References

[1] Accelrys. *CTfile Formats*. Accelrys, 2011. http://download.accelrys.com/freeware/.

[2] L. Victor Allis, Maarten van der Meulen, and H. Jaap van den Herik. Proof-number search. *Artif. Intell.*, 66(1):91–124, 1994.

[3] Carlos Areces, Facundo Bustos, Martín Dominguez, and Jörg Hoffmann. Optimizing planning domains by automatic action schema splitting. In Steve Chien, Minh Binh Do, Alan Fern, and Wheeler Ruml, editors, *ICAPS*. AAAI, 2014.

[4] Francis Carey and Robert Giuliano. *Organic chemistry*. McGraw-Hill, 8th edition, 2011.

[5] Michael Cashmore. *Planning as Quantified Boolean Formulae*. PhD thesis, Department of Computer Science, University of Strathclyde, UK, 2013.

[6] Michael Cashmore, Maria Fox, and Enrico Giunchiglia. Partially grounded planning as quantified boolean formula. In Daniel Borrajo, Subbarao Kambhampati, Angelo Oddi, and Simone Fratini, editors, *Proceedings of the Twenty-Third International Conference on Automated Planning and Scheduling, ICAPS 2013, Rome, Italy, June 10-14, 2013*. AAAI, 2013.

[7] ChemAxon. JChem software. http://www.chemaxon.com, accessed on Nov 9 2013.

[8] William Lingran Chen. Chemoinformatics: Past, present, and future. *Journal of Chemical Information and Modeling*, 46(6):2230–2255, 2006. PMID: 17125167.

[9] Anthony Cook, A. Peter Johnson, James Law, Mahdi Mirzazadeh, Orr Ravitz, and Aniko Simon. Computer-aided synthesis design: 40 years on. *Wiley Interdisciplinary Reviews: Computational Molecular Science*, 2(1):79–107, 2012.

[10] E. J. Corey and W. Todd Wipke. Computer-assisted design of complex organic syntheses. *American Association for the Advancement of Science*, 166(3902):178–192, 1969.

[11] Elias James Corey. Nobel lecture: The logic of chemical synthesis: Multistep synthesis of complex carbogenic molecules, 1990.

[12] Elias James Corey and Xue-Min Cheng. *The Logic of Chemical Synthesis*. Wiley-Interscience, 1989.

[13] Stefan Edelkamp and Jörg Hoffmann. PDDL2.2: The Language for the Classical Part of the 4th Intern. Planning Competition. Technical Report 195, Universität Freiburg, Institut für Informatik, 2004.

[14] Stefan Edelkamp and Shahid Jabbar. Cost-optimal external planning. In *Proceedings, The Twenty-First National Conference on Artificial Intelligence and the Eighteenth Innovative Applications of Artificial Intelligence Conference, July 16-20, 2006, Boston, Massachusetts, USA*, pages 821–826. AAAI Press, 2006.

[15] Shinsaku Fujita. Description of organic reactions based on imaginary transition structures. *Journal of Chemical Information and Computer Sciences*, 26(4):205–242, 1986.

[16] Shinsaku Fujita. Numbers of alkanes and monosubstituted alkanes. *Bull. Chem. Soc. of Japan*, 83(1):1–18, 2010.

[17] Johann Gasteiger, Matthias Pförtner, Markus Sitzmann, Robert Höllering, Oliver Sacher, Thomas Kostka, and Norbert Karg. Computer-assisted synthesis and reaction planning in combinatorial chemistry. *Perspectives in Drug Discovery and Design*, 20:245–264, 2000. 10.1023/A:1008745509593.

[18] Hector Geffner and Blai Bonet. A concise introduction to models and methods for automated planning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 7(2):1–141, 2013.

[19] H. Gelernter, N. S. Sridharan, H. J. Hart, S. C. Yen, F. W. Fowler, and H. J Shue. The discovery of organic synthetic routes by computer. *Topics Curr. Chem.*, 41:113, 1973.

[20] H. K. Gelernter, A. F. Sanders, D. L. Larsen, K. K. Agarwal, R. H. Boivie, G. A. Spritzer, and J. E Searleman. Empirical explorations of SYNCHEM. *Science*, 19:1041, 1977.

[21] Alfonso Gerevini, Yannis Dimopoulos, Patrik Haslum, and Alessandro Saetti. The 5th International Planning Competition: Deterministic part. http://ipc5.ing.unibs.it/, 2006.

[22] Alfonso Gerevini, Alessandro Saetti, Ivan Serina, and Paolo Toninelli. 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.

[23] Stephen Hanessian. Man, machine and visual imagery in strategic synthesis planning: Computer-perceived precursors for drug candidates. *Curr. Opin. Drug Discovery Dev.*, 8:798, 2005.

[24] Abraham Heifets. Benchmark problems, 2012. http://www.cs.toronto.edu/~aheifets/ChemicalPlanning/.

[25] Abraham Heifets and Igor Jurisica. Construction of new medicines via game proof search. In Jörg Hoffmann and Bart Selman, editors, *AAAI*, pages 1564–1570. AAAI Press, 2012.

[26] Malte Helmert. The Fast Downward Planning System. *J. Artif. Intell. Res.(JAIR)*, 26:191–246, 2006. http://hg.fast-downward.org.

[27] Malte Helmert. Concise finite-domain representations for PDDL planning tasks. *Artif. Intell.*, 173(5-6):503–535, 2009.

[28] J. B. Hendrickson, D. L. Grier, and A. G Toczko. A logic-based program for synthesis design. *J. Am. Chem. Soc.*, 107(18):5228–5238, 1985.

[29] Jörg Hoffmann and Bernhard Nebel. The FF planning system: Fast plan generation through heuristic search. *J. Artif. Intell. Res. (JAIR)*, 14:253–302, 2001.

[30] Ruoyun Huang, Yixin Chen, and Weixiong Zhang. A novel transition based encoding scheme for planning as satisfiability. In Maria Fox and David Poole, editors, *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2010, Atlanta, Georgia, USA, July 11-15, 2010*, pages 89–94. AAAI Press, 2010.

[31] Ruoyun Huang, Yixin Chen, and Weixiong Zhang. SAS+ planning as satisfiability. *JAIR*, 43:293–328, 2012.

[32] Shahid Jabbar. *External memory algorithms for state space exploration in model checking and action planning*. PhD thesis, Dortmund University of Technology, 2008.

[33] Craig James, Dave Weininger, and Jack Delany. *Daylight Theory Manual Ver. 4.9 (08/01/11)*. http://www.daylight.com/dayhtml/doc/theory/index.html, 2011.

[34] Philip Judson. *Knowledge-Based Expert Systems in Chemistry: Not Counting on Computers*. RSC Theoretical and Computational Chemistry. Royal Society of Chemistry, 2009.

[35] Henry A. Kautz, David A. McAllester, and Bart Selman. Encoding plans in propositional logic. In Luigia Carlucci Aiello, Jon Doyle, and Stuart C. Shapiro, editors, *Proceedings of the Fifth International Conference on Principles of Knowledge Representation and Reasoning (KR'96), Cambridge, Massachusetts, USA, November 5-8, 1996.*, pages 374–384. Morgan Kaufmann, 1996.

[36] Henry A. Kautz and Bart Selman. Planning as satisfiability. In *ECAI*, pages 359–363, 1992.

[37] D. Krebsbach, H. Gelernter, and S. M Sieburth. Distributed heuristic synthesis search. *J. Chem. Inf. Comput. Sci.*, 38(4):595–604, 1998.

[38] James Law, Zsolt Zsoldos, Aniko Simon, Darryl Reid, Yang Liu, Sing Yoong Khew, A. Peter Johnson, Sarah Major, Robert A. Wade, and Howard Y. Ando. Route designer: A retrosynthetic analysis tool utilizing automated retrosynthetic rule generation. *Journal of Chemical Information and Modeling*, 49(3):593–602, 2009.

[39] Qiang Lu, You Xu, Ruoyun Huang, and Yixin Chen. The Roamer planner: Random walk assisted

best-first search. In *7th International Planning Competition*, pages 73–76, 2011. http://staff.ustc.edu.cn/~qianglu8/software/seq-sat-roamer.tar.gz.

[40] Arman Masoumi and Mikhail Soutchanski. 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*, pages 35–44. AAAI Press, 2012.

[41] Arman Masoumi, Mikhail Soutchanski, and Andrea Marrella. Organic synthesis as artificial intelligence planning. In Adrian Paschke, Albert Burger, Paolo Romano, M. Scott Marshall, and Andrea Splendiani, editors, *Proceedings of the 6th International Workshop on Semantic Web Applications and Tools for Life Sciences, Edinburgh, UK, December 10, 2013.*, volume 1114 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2013.

[42] Drew McDermott. The 1998 AI planning systems competition. *AI Magazine*, 21:35–55, 2000.

[43] OEIS. A000602: Number of n-node unrooted quartic trees; number of $n$-carbon alkanes $C_nH_{2n+2}$ ignoring stereoisomers, 2014. http://oeis.org/A000602.

[44] Bernardus Cornelis Ridder. *Lifted Heuristics: Towards more scalable Planning Systems*. PhD thesis, Kings College, Department of Informatics, London, UK, 2014.

[45] Bram Ridder and Maria Fox. Heuristic evaluation based on lifted relaxed planning graphs. In *ICAPS*, 2014.

[46] Jussi Rintanen. Planning as satisfiability: Heuristics. *Artif. Intell.*, 193:45–86, 2012.

[47] Nathan Robinson, Charles Gretton, Duc Nghia Pham, and Abdul Sattar. Sat-based parallel planning using a split representation of actions. In Alfonso Gerevini, Adele E. Howe, Amedeo Cesta, and Ioannis Refanidis, editors, *Proceedings of the 19th International Conference on Automated Planning and Scheduling, ICAPS 2009, Thessaloniki, Greece, September 19-23, 2009*. AAAI, 2009.

[48] Akio Tanaka, Hideho Okamoto, and Malcolm Bersohn. Construction of functional group reactivity database under various reaction conditions automatically extracted from reaction database in a synthesis design system. *Journal of Chemical Information and Modeling*, 50(3):327–338, 2010. PMID: 20187659.

[49] Sylvie Thiébaux, Jörg Hoffmann, and Bernhard Nebel. In defense of PDDL axioms. *Artificial Intelligence*, 168(1-2):38 – 69, 2005.

[50] Matthew H. Todd. Computer-aided organic synthesis. *Chem. Soc. Rev.*, 34:247–266, 2005.

[51] Wang Tunghwa, Burnstein Ilene, Corbett Michael, Ehrlich Steven, Evens Martha, Gough Alice, and Johnson Peter. Using a theorem prover in the design of organic syntheses. In T.H. Pierce and B. A. Hohne, editors, *Artificial Intelligence Applications in Chemistry*, volume 306, pages 244–257, Chicago, Illinois, Sept 8-13, 1985, 1986. American Chemical Society.

[52] G. E. Vléduts. Concerning one system of classification and codification of organic reactions. *Inf. Storage Retr.*, 1:117, 1963.

[53] G. E. Vléduts. Do we still need a classification of reactions? In Peter Willett, editor, *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*, pages 202–220, Aldershot, England, 1986. Gower Publishing Company.

[54] G. E. Vléduts and E. A. Geivandov. *Automated Information Systems for Chemistry (in Russain)*. Nauka, Moscow, 1974.

[55] Richard Waldinger. Achieving several goals simultaneously. Technical Note 107, AI Center, SRI International, July 1975. Available at http://www.ai.sri.com/pubs/files/763.pdf.

[56] Tunghwa Wang. *An Expert System for Organic Synthesis Using Automated Reasoning*. PhD thesis, Illinois Institute of Technology, Chicago, Illinois, USA, 1986.

[57] W. Todd Wipke, Glenn I. Ouchi, and S. Krishnan. Simulation and evaluation of chemical synthesis - SECS: An application of artificial intelligence techniques. *Artif. Intell.*, 11(1-2):173–193, 1978.

[58] Nikolai S. Zefirov, Igor I. Baskin, and Vladimir A. Palyulin. SYMBEQ program and its application in computer-assisted reaction design. *Journal of Chemical Information and Computer Sciences*,

34(4):994–999, 1994.