CLAR paper

SMU CCLAW

July 7, 2021

Abstract

Your abstract.

1 Introduction

The purpose of this section is to give an account of the work we have been doing using Answer Set Programming to formalize and reason about legal rules. This approach is complementary to the one described before using SMT solvers. Here we will not go too much into the details of how various L4, language constructs map to the ASP formalisation. Our intention rather, is to present how some core legal reasoning tasks can be implemented in ASP while keeping the ASP representation readable, intuitive and respecting the idea of having an 'isomorphism' between the rules and the encoding. Going forward, our intention is to develop a method to compile L4 code to a suitable ASP representation like the one we shall now present. We first begin by formalizing the notion of what it means to 'satisfy' a rule set. We will do this in a way that is most amenable to ASP.

1.1 Formal Setup

Let the tuple $Config = (R, F, M, I)$ denote a *configuration* of legal rules. The set R denotes a set of rules of the form $pre\_con(r) \rightarrow concl(r)$. These are 'naive' rules with no information pertaining to any of the other rules in R. F is a set of positive atoms and predicates that describe facts of the legal scenario we wish to consider. M is a set of the binary predicates *despite*, *subject_to* and *strong_subject_to*. I is a collection of minimal inconsistent sets of positive atoms/predicates. Henceforth for a rule r, we may write C_r for it's conclusion $Concl(r)$

Throughout this document, whenever we use an uppercase or lowercase letter (like r, r_1, R etc.) to denote a rule that is an argument,in a binary predicate, we mean the unique numeric or alpha-numeric rule id associated with that rule. The binary predicate $legally\_valid(r, c)$ intuitively means that the rule r in R enforces conclusion c. The unary predicate $is\_legal(c)$ intuitively means that c legally holds. The predicates *despite*, *subject_to* and *strong_subject_to* all cause some rules to override others. Their precise semantics will be given next.

1.2 Semantics

A set S of is_legal and $legally\_valid$ predicates is called a *model* of $Config = (R, F, M, I)$, if and only if

A1) $\forall f \in F$ $is\_legal(f) \in S$.

A2) $\forall r \in R$, if $legally\_valid(r, C_r) \in S$. then $S \models is\_legal(pre\_con(r))$ and $S \models is\_legal(C_r)$ (this is a slight abuse of notation. Explain later?)

A3) $\forall c$, if $is\_legal(c) \in S$, then either $c \in F$ or there exists $r \in R$ such that $legally\_valid(r, C_r) \in S$ and $c = C_r$.

A4) $\forall r_1, r_2 \in R$, if $despite(r_1, r_2) \in M$ and $S \models is\_legal(pre\_con(r_2))$, then $legally\_valid(r_1, C_{r_1}) \notin S$

A5) $\forall r_1, r_2 \in R$, if $strong\_subject\_to(r_1, r_2) \in M$ and $legally\_valid(r_1, C_{r_1}) \in S$, then $legally\_valid(r_2, C_{r_2}) \notin S$

A6) $\forall r_1, r_2 \in R$ if $subject\_to(r_1, r_2) \in M$, and $legally\_valid(r_1, C_{r_1}) \in S$ and there exists a minimal conflicting set $i \in I$ such that $is\_legal(C_{r_1}) \in i$, $i \not\subseteq S$, and $i \subseteq S \cup \{is\_legal(C_{r_2})\}$, then $legally\_valid(r_2, C_{r_2}) \notin S$.

A7) $\forall r \in R$, if $S \models pre\_con(r)$, but $legally\_valid(r, C\_r) \notin S$, then it must be the case that at least one instance of A4 or A5 or A6 holds in which r is the 'lower priority rule'. (Maybe explicitly state what this means?) .

1.3 Some remarks on axioms A1-A7

Before we proceed let us give some informal intuition behind some of the axioms and their intended effects. A1 says that all facts in F automatically gain legal status. The set F represents indisputable facts about the legal scenario we are considering. A2 says that if a conclusion is enforced by a rule then both the pre-condition and conclusion of that rule must have legal status. Note that it is not enough if simply the conclusion has legal status as more than one rule may enforce the same conclusion or the conclusion may be a fact, so we want to know exactly which rules are 'in force' as well as their conclusions. A3 says that anything that has legal status must either be a fact or be enforced by the rules. A4 to A6 describe the semantics of the three modifiers. The intuition for the three modifiers here is that with despite, once the precondition of the higher priority rule is true, it invalidates the lower priority rule regardless of whether the higher priority rule actually comes into effect. With 'strong subject to', once the higher priority rule is in effect, then it invalidates the lower priority rule. With 'subject to', The higher priority rule has to be in effect and it has to contradict the lower priority rule

for the lower priority rule to be invalidated. We will give examples later on to illustrate these modifiers. A7 says essentially that A4-A6 represent the only ways in which a rule whose pre-condition is true may nevertheless not be in effect, and any rule whose precondition is satisfied and is not invalidated directly by some instance of A4-A6 must be in effect.

2 Non-existence of models

Note that there may be configurations for which no models exist. This is most easily seen in the case where there is only one rule, the pre-condition of that rule is given as fact, and the rule is strongly subject to itself.

3 ASP encoding

Here is an ASP encoding scheme given a configuration $Config = (R, F, M, I)$ of legal rules.

```
% For any f in F, we have:
is_legal(f).

% Any rule r in R is encoded using the general schema:
according_to(r,C_r):-is_legal(pre_con(r)).

% Say {a,b,c} is a minimal inconsistent set in I, then this would get encoded as:
opposes(a,b):-is_legal(c)
opposes(a,c):-is_legal(b).
opposes(b,c):-is_legal(a).
% The above is done for every minimal inconsistent set. A pair from the set forms the opposes
%predicate and the rest of the elements go in the body.

opposes(X,Y):-opposes(Y,X).



% Encoding for 'despite'
defeated(R,C):-according_to(R,C),according_to(R1,C1),despite(R,R1).

% Encoding for 'subject_to'
defeated(R,C):-according_to(R,C),legally_valid(R1,C1),opposes(C,C1),subject_to(C1,C).

% Encoding for 'strong_subject_to'
defeated(R,C):-according_to(R,C),legally_valid(R1,C1),strong_subject_to(C1,C).
```

```
legally_valid(R,C):-according_to(R,C),not defeated(R,C).

is_legal(C):-legally_valid(R,C).
```

3.1 Lemma

For a configuration $Config = (R, F, M, I)$, let the above encoding be the program ASP_{Config}. Then the sets of *is_legal* and *legally_valid* predicates from answer sets of ASP_{Config} correspond exactly to the *models* of $Config$.

Proof sketch The definition of $defeated(R, C)$ that involves *despite* encodes exactly A4, the definition of $defeated(R, C)$ that involves *strong_subject_to* encodes exactly A5, and the way the *opposes* predicates are defined together with the remaining definition of $defeated(R, C)$ encodes exactly A6. Other than rules that are invalidated this way, all $according\_to(R, C)$ predicates get turned into $legally\_valid(R, C)$ predicates (A7), and the only way to derive a $is\_legal(C)$ predicate is if C is a fact or C has been enforced by a rule. (First and last lines of encoding)

3.2 Comments/Thoughts

It is not entirely clear to me how to unify this approach with Martin's approach to the semantics of L4/where this stuff fits in exactly. However I think it can be done? It seems that our notions of what a rule is match well. Class hierarchies can be incorporated using rules like $is\_truck(X) \rightarrow is\_vehicle(X)$. In this set-up this information along with minimal inconsistent sets can all be stored in the parameter I from the tuple. So perhaps we could rename it T for background theory?

The minimal inconsistent sets in I seem to somewhat loosely correspond to 'assertions' in Martin's set up although they seem to mean different things. In this set up, they are meant to encode basic domain knowledge about things that contradict each other. But I think they can also be used as a tool to check properties of the program.

Our notions of 'subject to' are different but 'strong subject to' here seems close to Martin's 'subject to'.

I'm not sure how to go about properly proving the lemma.

Need to do examples to illustrate the different kinds of modifiers.

References