Default Reasoning in CHR$^\vee$

Marcos Aurélio[1,2], François Fages[2], Jacques Robin[1]

[1] Universidade Federal de Pernambuco, Recife, Brazil
[2] INRIA, Rocquencourt, France

Abstract. CHR$^\vee$ has emerged as a versatile knowledge representation language, usable for an unparalleled variety of automated reasoning tasks: constraint solving, optimization, classification, subsumption, classical deduction, abduction, truth-maintenance, belief revision, belief update and planning. In this paper, we add default reasoning to this list, by showing how to represent default logic theories in CHR$^\vee$. We then discuss how to leverage this representation together with the well-know correspondence between default logic and *Negation As Failure* (NAF) in logic programming, to propose an extension CHR$^{\vee,naf}$ of CHR$^\vee$ allowing NAF in the rule heads.

1 Introduction

CHR$^\vee$ [1] is a first-order, relational rule language that incorporates forward chaining of conditional rewrite rules and guarded production rules, with logic programming's backtracking search of disjunctive alternatives in the rules' right-hand side (called "body" in CHR$^\vee$).

It was initially conceived to declaratively implement constraint solving tasks by harmoniously integrating three techniques widely used for this task: (a) constraint simplification using conditional rewrite rules (called "simplification rules" in CHR$^\vee$), (b) constraint propagation using guarded production rules (called "propagation rules" in CHR$^\vee$), and (c) finite domain constraint search using disjunctive rules (either simplification or propagation rules).

This integration turned out to be powerful enough to use CHR$^\vee$ for a surprisingly wide variety of other reasoning tasks beyond constraint solving: classical deduction [8], description logics, concept subsumption and individual classification [7], abduction [2], truth-maintenance [15], belief revision [9], belief update and planning [13].

In this paper, we first show how to add default reasoning to the list of automated reasoning tasks that can be performed by reusing a CHR$^\vee$ inference engine. Our proposal is based on a mapping of a Default Logic Theory[12] into a CHR$^\vee$ rule base. We then leverage this mapping, together with the well-known correspondence between default reasoning and logic programming with Negation As Failure (NAF) [6], to propose an extension CHR$^{\vee,naf}$ of CHR$^\vee$ that allows the `naf` connective in rule heads. We also explain how our CHR$^{\vee,naf}$ proposal semantically differs from the CHR$^\neg$ Negation As Absence (NAA) proposal [14] and why we believe its applicability is wider.

The contributions of this paper are (a) mapping of default logic formulas into CHR$^\vee$ bases, and (b) leveraging it to extend CHR$^\vee$ with NAF. This is very significant due to (a) the pervasive utility of default reasoning and NAF in artificial intelligence applications, and (b) their well-studied relations with other forms of non-monotonic reasoning abduction [10], truth-maintenance [10], belief revision [11] and inheritance with overriding [16]. Each of these reasoning tasks must be carried out by any agent that acts in a partially observable environment, the most common case in practical applications.

The rest of this paper is organized as follows. In the next sections, we briefly review in turn the syntax and semantics of Default Logic and CHR$^\vee$. In section four we present our mapping of the former to the latter. In section five, we show how to leverage such mapping to extend CHR$^\vee$ with NAF. In section six, we discuss related work in non-monotonic reasoning in CHR$^\vee$, before concluding in section seven.

2 Default Logic

Default Logic [3][5][12] formalizes the reasoning of an agent in a partially observable environment, where it misses some volatile knowledge, typically the truth value of a fluent[3], that is essential to choose its next action. In such situation, the agent needs to base its choice on some default hypothesis about the truth value of that fluent. While this hypothesis must be consistent with the agent's current volatile knowledge about the current state of the environment, it can nevertheless be deductively unsound and thus subject to revision upon subsequent deduction from reliable new sensor information of contradictory volatile knowledge.

Such reasoning cannot be appropriately formulated directly in Classical First-Order Logic (CFOL) due to the knowledge monotonicity assumption of this formalism. This is illustrated by the following example:

Example 1. Let us assume that we want to represent the following piece of knowledge: *Birds typically fly, Penguins and Albatrosses are birds, Penguins do not fly and Tux is a Penguin.*

The CFOL formula $\forall x((Bird(x) \rightarrow Flies(x)) \wedge (Penguin(x) \rightarrow Bird(x)) \wedge Penguin(tux) \wedge (Penguin(x) \rightarrow \neg Flies(x)) \wedge (Albatross(x) \rightarrow Bird(x)) \wedge \neg(Penguin(x) \wedge Albatross(x)))$ does not properly represent such knowledge because it entails $\bot$ due to the inability of pure deduction in CFOL to retract the conclusion $Flies(tux)$ entailed by the first three clauses of the formula in the light of the more specific conclusion $\neg Flies(tux)$ entailed by the third and fourth clauses. The *core* problem of CFOL is the inability to represent rules with exceptions with the only two quantifiers of CFOL: universal and existential. Default logic extends CFOL with default inference rules that capture such rules with exception. It is formally defined below.

[3] A property of the environment that changes over time or due to the actions executed by the agent.

Definition 1 (Default Logic Theory). *A Default Logic Theory is a tuple* $\langle D, W \rangle$ *where* D *is the set of default rules and* W *is a set of CFOL formulas. Each default rule assumes the following form:*

$$\frac{\alpha : \beta}{\gamma}$$

where α (*prerequisite*), β (*justification*) and γ (*conclusion*) are CFOL formulas. The intended meaning of this default rule is:

If α *is entailed by the current knowledge base* ($KB \models \alpha$) *and* β *is consistent with the current knowledge base* ($KB \wedge \beta \nvDash \bot$) *then* γ *can be assumed.*

The *extension* ε of a Default Theory is the maximal set of formulas that can be derived and assumed by default from it. This concept is formally defined in the Definition 3. Notice that a Default Theory may have one, many or no extension at all.

Definition 2 (Deductive Closure). *Let* T *be a set of CFOL formulas. The* Deductive Closure *of* T *is a set* $Th(T)$ *such that* $T \subseteq Th(T)$ *and for each* $p \in Th(T)$, *if* $p \models q$ *then* $q \in Th(T)$.

Definition 3 (Extension of a Default Theory). *We define* ε *as an extension for the default theory* $\langle D, W \rangle$ *if and only if it satisfies the following equations:*

$$E_0 = W$$

and for $i > 0$,

$$E_{i+1} = Th(E_i) \cup \left\{ \gamma | \frac{\alpha : \beta}{\gamma} \in D, \alpha \in E_i, \neg\beta \notin \varepsilon \right\}$$

and,

$$\varepsilon = \bigcup_{i=0}^{\infty} E_i$$

Example 2. Let us show how to model the knowledge in the Example 1 as a Default Theory. At first, the default rules:

$$D = \left\{ \frac{Bird(x) : Flies(x)}{Flies(x)}, \frac{Bird(x) : Penguin(x)}{Penguin(x)}, \frac{Bird(x) : Albatross(x)}{Albatross(x)} \right\}$$

Notice that, when we state this knowledge as default rules, we emphasize what is assumed by hypothesis. We are now going to represent the logical formulas of our theory:

$$W = \{Penguin(tux), Penguin(x) \rightarrow \neg Flies(x), \neg(Penguin(x) \wedge Albatross(x))\}$$

Two of the possible extensions for this theory are:

$$\varepsilon_1 = W \cup \{Flies(x), Albatross(x)\}$$

and,

$$\varepsilon_2 = W \cup \{\neg Flies(x), Penguin(x)\}$$

2.1 NAF as Default Logics

In this section we show how to express NAF by means of Default Theories. This negation is different from the usual CFOL one. In order to avoid misinterpretations we utilize the symbol naf for negation as failure and *cneg* for classical logical negation.

Take the following rules:

$$p \leftarrow cneg(q).$$

$$p' \leftarrow naf(q).$$

In the first case, p can be proved only if q can be *proved* to be false. In the second one, p' can be assumed to be true if q cannot be proved to be true. The difference relies on the fact that q may be true, but unknown (i.e., it may not be possible to deduce it). In the first example it is not possible to deduce p (because it is not possible prove q false), but in the second example it is possible to deduce p' (because it is not possible to prove q true).

In [3], Grigoris Antoniou shows how to model NAF as Default Theories in a natural way. The idea is to add a default rule like the following one for each ground fact ϕ:

$$\frac{: naf\phi}{naf\phi}$$

This means that if the hypothesis $naf\phi$ is consistent (in other words, if ϕ can't be proved), $naf\phi$ can be assumed.

3 CHR$^\vee$

Constraint Handling Rules with Disjunction (CHR$^\vee$) [1] is a first-order, relational rule language for writing Constraint Solvers.

There are two kinds of constraints: the *user defined* and the *built-ins*. The first set is formed by the constraints whose semantics is given by the set of rules

and the second set is formed by the constraints whose semantics is provided by the inference engine.

There are three kinds of rules in CHR$^\vee$: simplification, propagation and simpagation. They can be respectively described as follows:

$$r@H_r \Leftrightarrow G|B.$$

$$s@H_k \Rightarrow G|B.$$

$$t@H_k \backslash H_r \Leftrightarrow G|B.$$

In this example, `r`, `s` and `t` are identifiers for the rules and can be omitted. H_r and H_k are the heads of the rules. More specifically, H_k are the *kept heads*, which are kept in the constraint store; and H_r are the *removed heads*, which are removed from it. G is the guard and B is the body. If the guard is `true`, it can be omitted.

The abstract operational semantics for CHR$^\vee$ is defined as a transition system. A CHR$^\vee$ state is a disjunction of one of more *subgoals* which are conjunctions of user defined constraints, built-ins or disjunctions. A state is called final if no transition is applicable or all of its subgoals are inconsistent (in this case, it is called *failed*). For more details see [1].

The following diagram presents the transition rules of CHR$^\vee$:

Solve
If $CT \models \forall(S \Leftrightarrow S')$ and S' is the normal form of S
then $S \mapsto_P S'$

Propagate
If $(H \Rightarrow G|B)$ is a fresh variant of a rule with variables $\bar{x}$
and $CT \models \forall(S \rightarrow \exists\bar{x}(H = H' \wedge G)$
then $(H' \wedge S) \mapsto_P (H = H' \wedge B \wedge G \wedge H' \wedge S)$

Simplify
If $(H \Leftrightarrow G|B)$ is a fresh variant of a rule with variables $\bar{x}$
and $CT \models \forall(S \rightarrow \exists\bar{x}(H = H' \wedge G)$
then $(H' \wedge S) \mapsto_P (H = H' \wedge B \wedge G \wedge S)$

Simpagate
If $(H_k \backslash H_R \Leftrightarrow G|B)$ is a fresh variant of a rule with variables $\bar{x}$
and $CT \models \forall(S \rightarrow \exists\bar{x}(H_k = H'_k \wedge H_R = H'_R \wedge G)$
then $(H'_k \wedge H'_R \wedge S) \mapsto_P (H_k = H'_k \wedge H_R = H'_R \wedge B \wedge G \wedge H'_k \wedge S)$

Split
$(S_1 \vee S_2) \wedge S \mapsto_P (S_1 \wedge S) \vee (S_2 \wedge S)$

4 Describing Default Logic Theories in CHR$^\vee$

In this Section we describe our approach to describing Default Theories as CHR$^\vee$ rule bases. It consists of mapping each default rule into two sets of propagation rules. Let us take the following default rule:

$$\frac{\alpha : \beta}{\gamma}$$

Without loss of generality, consider α and γ as being in the Conjunctive Normal Form (CNF) and β as being in the Disjunctive Normal Form (DNF) [4]:

$$\alpha = (\alpha_{1,1} \wedge \ldots \wedge \alpha_{n,1}) \vee \ldots \vee (\alpha_{1,m} \wedge \ldots \wedge \alpha_{n,m})$$

$$\beta = (\beta_{1,1} \vee \ldots \vee \beta_{n,1}) \wedge \ldots \wedge (\beta_{1,m} \vee \ldots \vee \beta_{n,m})$$

$$\gamma = (\gamma_{1,1} \wedge \ldots \wedge \gamma_{n,1}) \vee \ldots \vee (\gamma_{1,m} \wedge \ldots \wedge \gamma_{n,m})$$

Definition 4 (Search Rules). *Given a default rule r as above, we define search(r) as the set containing the following rules:*

```
r1 @ α11, ..., αn1 ==> (r, γ) ; true.
...
rn @ α1m, ..., αnm ==> (r, γ) ; true.
```

The intended meaning of these rules is: if α, or any of its disjunctive components, is in the Constraint Store (and is therefore entailed by it), we have two options:

- assume β and add γ to the Constraint Store,
- do not assume β.

The new constraint `r` has exactly the meaning of *β is assumed by default.* This can be accomplished by adding β to the constraint store. However, when dealing with simplification and simpagation rules, part of β may be removed by some simplification or simpagation rule and it might not be possible to prove its falsehood in the future.

Definition 5 (Integrity Rules). *Given a default rule r as above, we define integrity(r) as the set containing the following rules:*

```
s1 @ r, cneg_β11, ..., cneg_βn1 ==> false.
...
sn @ r, cneg_β1m, ..., cneg_βnm ==> false.
```

[4] Notice that we employ Abdennadher's notation for negation as defined in [2], in which a negated constraint $cneg(\phi)$ is represented by a new constraint $cneg\_\phi$ and a integrity constraint $\phi, cneg\_\phi \Rightarrow false$.

The intended meaning of this set of rules is: if $cneg\_\beta$ can be proved, or any of its components, and β has been assumed, then the store is inconsistent.

The idea behind this solution is to use CHR$^\vee$ as a platform for *searching* for an extension for the Default Theory. For example, let us return to the Example 1. The result of the application of the transformation to this problem is:

```
r1 @ bird(X) ==> (r1(X), flies(X)) ; true.
r2 @ bird(X) ==> (r2(X), penguin(X)) ; true.
r3 @ bird(X) ==> (r3(X), albatross(X)) ; true.

s1 @ r1(X), cneg_flies(X)     ==> false.
s2 @ r2(X), cneg_penguin(X)   ==> false.
s3 @ r3(X), cneg_albatross(X) ==> false.
```

Let us feed the CHR$^\vee$ engine with the following set of rules and initial constraint store:

```
penguin(X) ==> cneg_flies(X).
penguin(X), albatross(X) ==> false(X).

query: bird(tux).
```

The following set of final states is going to be obtained (omitting the new constraints added by the transformation):

```
S1 = {  bird(tux), albatross(tux), flies(tux)  }

S2 = {  bird(tux), albatross(tux)  }

S3 = {  bird(tux), penguin(tux), cneg_flies(tux)  }

S4 = {  bird(tux), flies(tux)  }

S5 = {  bird(tux) }
```

Notice that each store corresponds to a set of assumed hypotheses, ranging from two hypotheses in `S1` and `S3` to no hypothesis in `S5`.

We are now going to demonstrate that our approach successfully computes all correct extensions, at least for a restricted set of Default Theories. We call this restricted set of *Propositional CHR Propagation Restricted Default Theories*.

Definition 6 (Propositional CHR Default Theory). *A default theory $\langle D, W\rangle$ is said to be a* Propositional CHR Default Theory*, if*

- *D is composed of default rules of the form, where α and γ are in the CNF and β is in the DNF, and are propositional CFOL formulas:*

$$\frac{\alpha : \beta}{\gamma}$$

– *W is composed of a conjunction of:*
  - *A set of atomic constraints,*
  - *A set of logical rules of the form $H \wedge G \rightarrow B$ or $G \rightarrow (H \leftrightarrow B)$, equivalent to CHR propagation and simplification rules without disjunctions, respectively.*
  - *For each constraint ϕ there is a rule of the form: $\phi \wedge cneg\_\phi \rightarrow \bot$*

Definition 7 (Propositional CHR Propagation Restricted Default Theories). *A default theory $\langle D, W\rangle$ is said to be* Propositional CHR Propagation Restricted *, if and only if it is a Propositional CHR Default Theory and the rules in W are equivalent to non-disjunctive CHR propagation rules.*

Our notion of *equivalence* of CHR states and Default Theories extensions is outlined by the Definition 8. It captures the fact that the new constraints introduced by our approach do not change the meaning of a state.

Definition 8 (Equivalence of States and Theories). *Let $\langle W, D\rangle$ be a Propositional CHR Propagation Restricted Default Theory, and let W be the conjunction of the constraints in a $CHR^{\vee}$ state S and P a set of CHR rules. We say that S is equivalent to W if W contains the logical meanings of the rules in P, the constraints in S and no other instance of a constraint appearing in P or in S.*

Theorem 1. *Let $\langle D, W\rangle$ be a Propositional CHR Propagation Restricted Default Theory. Let W be the conjunction of the initial goal S and the set of CHR rules P, and R the set obtained by transforming the default rules in D into CHR rules. For every extension ε of $\langle D, W\rangle$, if E_i is equivalent to a subgoal S'_g of some state S' (such that $S \mapsto_{P\cup R} \ldots \mapsto_{P\cup R} S'$ is a finite derivation), then there exists an state S'' such that E_{i+1} is equivalent to a subgoal S''_g of S'' and $S' \mapsto_{P\cup R} \ldots \mapsto_{P\cup R} S''$ is a finite derivation for it.*

Proof (Sketch). If ε is an extension and the default rule $r = \frac{\alpha:\beta}{\gamma}$ is applied between the step E_i and E_{i+1}. Since the extension exists, β is consistent with it. It's easy to see that we can divide the derivation between S' and S'' into two steps: (i) compute the deductive closure of S'_g and (ii) execute some of the rules in $search(r)$. Applying all the derivation steps to the subgoal S'_g will lead us to a state S'' containing a subgoal S''_g which is equivalent to E_{i+1}.

□

Theorem 2 (Completeness). *Let $\langle D, W\rangle$ be a Propositional CHR Propagation Restricted Default Theory. Let W be the conjunction of the initial goal S and the set of CHR rules P, and R the set obtained by transforming the default rules in D into CHR rules. For each non-failed derivation $S \mapsto^*_{P\cup R} S_f$, all extensions ε of $\langle D, W\rangle$ are subgoals of S_f.*

Proof. By contradiction. Let us suppose there exists an extension ε which is not a subgoal of S_f. By definition, $\varepsilon = \bigcup_{i=0}^{\infty} E_i$. By Theorem 1, it follows easily that every E_i should be equivalent to a subgoal of S_f, and thus ε should also be a subgoal of S_f.

□

Theorem 3 (Weak Correctness). *Let $\langle D, W\rangle$ be a Propositional CHR Propagation Restricted Default Theory. Let W be the conjunction of the initial goal S and the set of CHR rules P, and R the set obtained by transforming the default rules in D into CHR rules. For each non-failed derivation $S \mapsto^*_{P\cup R} S_f$, every non-failed subgoal of S_f is equivalent to a subset of an extension ε of $\langle D, W\rangle$.*

Proof (Sketch). By induction on the derivation length. The initial constraint store is equivalent to a subset of every extension, by definition. By the proof of the Theorem 1 it is easy to verify that each transition in $S \mapsto^*_{P\cup R} S_f$ is one step of the computation of a deductive closure or in the execution of a default rule between some step E_i and E_{i+1} in the derivation of an extension ε.

□

If we try to extend these results to less restricted versions of the Theorems 2 and 3 we are going to see that both properties are going to be lost, mainly due to the fact that the simpagation and simplification actually remove part of the state. Therefore, the obtained subgoals are not going to be complete extensions anymore.

Another possible extension is allowing disjunctive bodies. In this case, an extension is not going to correspond to a subgoal, but to a set of subgoals, which can be easily computed, each subgoal contains an extra constraints for the assumed hypothesis: each explanation is the disjunction of the subgoals relying on the same hypotheses.

5 CHR$^{\vee,naf}$: Extending CHR$^{\vee}$ with NAF in the rule heads

In this Section, we are going to present CHR$^{\vee,naf}$, an extension for CHR with negated rule heads. In this extension, there are three kinds of rules: simpagation, propagation and simplification. The simpagation rules generalize all of them. Their general syntax is:

$$r@H_k \backslash H_r \backslash\backslash N_1|G_1 \backslash\backslash \ldots \backslash\backslash N_n|G_n \Leftrightarrow G|B.$$

In this example, r is an identifier for the rule, which can be omitted. H_k are the *kept heads*, which are kept in the constraint store when the rule fires. The H_r are the *removed heads*, which are removed when the rule fires. H_k and H_r are the *positive heads* of the rule, whereas $N_1, \ldots, N_n$ are the *negative heads*. All heads must contain only user defined constraints. $G_1, \ldots, G_n$ are the negated guards, and like the positive guard G, may be empty and in this case can be omitted.

Any variable introduced in a guard cannot be used in another guard, i.e., the guards can only use the variables appearing in the positive heads and in their corresponding negative head. The variables defined by the negative guards cannot appear in the rule body and all guards are composed only of built-in constraints.

If H_k is empty, it can be omitted (along with the following *backslash*) and the rule is called a *simplification* rule. If H_r is empty, it can also be omitted (along with the preceding *backslash*) and the sign $\Leftrightarrow$ is changed to $\Rightarrow$. This rule is called *propagation* rule. The negative heads cannot be empty. A rule is authorized to have from 0 to any number of negated heads.

Finally, B is the rule body, and is composed of a disjunction of conjunctions of user defined and built-in constraints. No variable defined in a negative head can appear in the rule body.

For now, let us consider the semantics of a rule such that as being:

$$\forall G \rightarrow (H_k \wedge H_r \wedge naf(\exists((N_1 \wedge G_1) \wedge \ldots \wedge (N_n \wedge G_n))) \leftrightarrow B)$$

In other words: *if the head is in the constraint store and there is no proof that the negated head is inconsistent with it, we can assume it by hypothesis.*

Example 3. Let us suppose we want to find the minimum value `X` for which there exists a `c(X)` in the constraint store.

The common approach is to do something like:

```
c(X) \ getMin(Min) <=> current(X,Min).
c(X) \ current(Current,Min) <=> X<Current | current(X,Min).
current(Current, Min) <=> Min = Current.
```

Notice that this solution explores the refined operational semantics of CHR and it is inherently *not* confluent. In a isolated piece of code like this these properties might not cause grave problems. However, in large rule bases, confluence problems may be much harder to solve. We want, as much as possible, to find confluent solutions to the problems.

In CHR$^{\vee,naf}$, this example is implemented much simply, by the means of following rule:

```
getMin(M), c(X) \\ c(Y) | Y < X ==> M = X.
```

The logical reading of this rule is:

$$\forall X, M(getMin(M) \wedge c(X) \wedge naf(\exists Y(c(Y) \wedge Y < X)) \rightarrow X = M)$$

5.1 Negation in Rule Heads as Default Reasoning

Now we show how to extract a Default Theory from a CHR$^{\vee,naf}$ rule base. Let us take the CHR$^{\vee,naf}$ rule from the Example 3:

```
getMin(M), c(X) \\ c(Y) | Y < X ==> X = M.
```

It is possible to translate the logical reading of this rule in the following default rule:

$$\forall X, Y, M \frac{getMin(M), c(X) : \neg(c(Y) \wedge Y < X)}{X = M}$$

From this example, it is possible to infer that every CHR$^{\vee,naf}$ propagation rule can be naturally translated into a default rule. The general pattern is that the following rule:

$$r@H \backslash\backslash N_1 | G_1 \backslash\backslash \ldots \backslash\backslash N_n | G_n \Rightarrow G | B.$$

generates the following default rule:

$$\forall \frac{H \wedge G : (\neg(N_1 \wedge G_1) \wedge \ldots \wedge \neg(N_n \wedge G_n))}{B}$$

Unfortunately, this translation does not account either for general simpagation rules or for simplification rules. What is missing is the capability of removing the constraints in the head from the constraint store.

This can be easily accomplished by mapping each simpagation or simplification rule into a pair of rules, one for propagating the body and another for removing the head.

For example, let us take the following simpagation rule:

```
r @ a(X) \ b(Y) <=> g(Z) | c.
```

It is possible to rewrite it into an equivalent pair of rules:

```
r1 @ a(X), b(Y) ==> g(Z) | s(X,Y,Z).
r2 @ s(X,Y,Z), b(Y) <=> c.
```

This pair of rules is equivalent to former one (in the sense of the Definition 8). Since the negated heads in a CHR$^{\vee,naf}$ rule base act like a precondition, this strategy can be extended to CHR$^{\vee,naf}$ programs. The idea is than translate each simpagation rule of the form:

$$r@H_k \backslash H_r \backslash\backslash N_1 | G_1 \backslash\backslash \ldots \backslash\backslash N_n | G_n \Leftrightarrow G | B.$$

to a pair of rules, a propagation and a simplification rule:

$$r_1@H_k, H_r \backslash\backslash N_1 | G_1 \backslash\backslash \ldots \backslash\backslash N_n | G_n \Rightarrow G | s.$$

$$r_2@s, H_r \Leftrightarrow B.$$

The next step is then, transforming the rule r_1 into a default rule, the result is:

$$\forall \frac{H_k \wedge H_r \wedge G : \neg(N_1 \wedge G_1) \wedge \ldots \wedge \neg(N_n \wedge G_n)}{s}$$

The following set of CHR$^\vee$ rules is obtained[5].

```
r21 @ Hk, Hr ==> G | (r, s) ; true.
r22 @ r, N1 ==> G1 | false.
...
r2n @ r, Nn ==> Gn | false.
```

Let us return to the Example 3. The complete transformed rule base is as follows:

```
 r1 @ getMin(M), c(X) ==> (r(X,M), X = M) ; true.
 r2 @ r(X,M), c(Y) ==> Y < X | false.
```

Let us suppose we feed the CHR$^\vee$ inference engine with the following goal:

```
c(3), c(9), getMin(M)
```

The two final Constraint Stores are:

```
S1 = { c(3), r(3,3), c(9), getMin(3) }
S2 = { c(3) , c(9), getMin(M) }
```

The first one is obtained by assuming `c(3)` as the minimum and the other one is obtained by assuming no constraint as the minimum.

6 Related Work

6.1 Abduction in CHR$^\vee$

In [10], Kakas et al show that Default Logics is a special case of Abduction. They say that the process of assuming hypotheses can be viewed as a form of abduction, where instances of defaults are the candidate abducibles.

In [2], Abdennadher explains how to utilize CHR$^\vee$ as a platform for Abductive Reasoning and presents a method for expressing abductive problems as CHR$^\vee$ rule bases.

In theory, it is possible to combine both approaches in order to reason about Default Theories in CHR$^\vee$. The main advantage of our approach is that it allows the addition of default rules to existing CHR$^\vee$ rule bases, while the hybrid approach combining [10] and [2] would require the translation of the existing rule bases into abductive problems and then the translation of these problems into CHR$^\vee$, which might not be trivial.

[5] Notice that we map former guards into guards in the new rules.

6.2 Comparing CHR$^{\neg}$ and CHR$^{\vee,naf}$

Both CHR$^{\neg}$ and CHR$^{\vee,naf}$ share the same syntax, but differ substantially in their semantics. The semantics for CHR$^{\neg}$ was based on the refined operational semantics for CHR defined in [4] and consisted of restricting the applicability of CHR rules to situations where no negated head were present and adding the notion of *Triggering on removal*, in which, a rule should also fire when a negated constraint is removed from the constraint store.

That semantics presented some undesirable features, which this one aims to overcome. The first of these effects is the lost logical reading, in the sense that, because of its essentially operational semantics, it is not anymore possible to map each rule into a logical formula. The present semantics brings back a logical reading to rules with negative heads, in the sense that each rule can be read as a *default rule*, where the pre-requisites are in the positive head, the justification is the negative head and the conclusion is the body.

Another undesired feature is the unexpected behavior for some programs. The operational semantics for CHR$^{\neg}$ turned out to lead to counter-intuitive results for some simple programs, as the one described in the Example 4.

Example 4 (Order). Under CHR$^{\neg}$, negatively occurring constraints have to be added in the right order. In the following rule base, everytime the first rule fires, the child is declared an orphan. The reason for that is the fact that when the constraint `child(C)` is added to the constraint store and the constraints `father(F,C)` and `mother(M,C)` have not yet been added, and thus, the second rule fires.

```
birth(C,F,M) <=> child(C), father(F,C), mother(M,C).
child(C) \\ father(_,C) \\ mother(_,C) ==> orphan(C).
```

To illustrate the difference between both semantics, let us suppose we try the following initial constraint store:

```
birth(a, b,c), child(e).
```

In this example, `a` is not orphan, but we don't know whether `e` is. We are going to obtain two final constraint stores:

```
S1 = { child(a), father(b, a), mother(c, a), child(e), orphan(e) }
S2 = { child(a), father(b, a), mother(c, a), child(e) }
```

The first one, assumes `e` to be an orphan, and the second one does not assume anything. In CHR$^{\neg}$, the final constraint store for this initial constraint store is:

```
S1 = { child(a), father(b, a),
       mother(c, a), child(e),
       orphan(a), orphan(e) }
```

This result is unexpected because `a` is clearly not orphan.

7 Conclusion

At this work, we confirmed the flexibility of the CHR$^\vee$ language by presenting it as a platform for Default Reasoning services. We defined an approach that permits us to rewrite Default Rules as CHR$^\vee$ propagation rules and reuse the built-in search capabilities of CHR$^\vee$ in order to find consistent sets of hypotheses that can be assumed in a given Default Theory.

We have also investigated how to leverage the correspondence between Default Logic and *Negation As Failure* (NAF) in order to propose an extension CHR$^{\vee,naf}$ for CHR$^\vee$ allowing negated constraints and guards in the rule head. We showed how this extension relate to CHR$^\neg$ [14], which employs an operational concept of negation: *Negation As Absence* (NAA).

We propose the following future works:

- **Triggering on Removal:** This is an important feature of CHR$^\neg$ which is not supported by CHR$^{\vee,naf}$ because it is not declarative. In order to allow this kind of reasoning we would need to employ some better-founded semantics for removal, like the one employed by Adaptive CHR$^\vee$ [15].

- **Complexity of Default Logics in CHR$^\vee$:** As pointed out by [5], the problem of enumerating all the extensions for a Default Theory has an exponential time complexity. This is easily shown by the fact that each possible hypothesis generates two possibilities: considering it and not considering it. Under this context, it is easy to notice that the number of states computed by the CHR$^\vee$ machine increases exponentially with the input size.

  In fact, only some of the returned states are complete extensions. For example, in the list of final states presented at the example in the Section 4, only the first one was really an extension. One of the future works is to develop an operational semantics taking a bias in the hypothesis into accounts, making it possible to prioritize the returned solutions.

  This new strategy will not be able to reduce its worst case complexity, but will improve its average time complexity.

- **Stronger Theoretical Results:** the proofs presented at the Section 2.1 cover only a very limited range of Default Theories (the Propositional CHR Propagation Restricted Default Theories). A future work is to extend this results, initially to Default Theories with variables and quantifiers and then to simpagation and simplification rules.

References

1. Slim Abdennadher. A Language for Experimenting with Declarative Paradigms. *Second Workshop on Rule-Based Constraint Reasoning and Programming*, 2000.

2. Slim Abdennadher. Rule-Based Constraint Programming: Theory and Practice. Technical report, Institut für Informatik, Ludwig-Maximilians-Universität München, July 2001.
3. Grigoris Antoniou. A tutorial on default logics. *ACM Computing Surveys*, 31(4):337–359, 1999.
4. Gregory J. Duck, Peter Stuckey, María García de la Banda, and Christian Holzbaur. The Refined Operational Semantics of Constraint Handling Rules. In *Proceedings of the 20th International Conference on Logic Programing (ICLP'04)*, pages 90–104, Saint-Malo, France, 2004. Springer Berlin / Heidelberg.
5. Thomas Eiter and George Gottlob. Semantics and complexity of abduction from default theories. In Chris Mellish, editor, *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence*, pages 870–877, San Francisco, 1995. Morgan Kaufmann.
6. François Fages. Consistency of Clark's Completion and Existence of Stable Models. *Methods of Logic in Computer Science*, pages 1:51–60, 1994.
7. Thom Frühwirt. Description Logic and Rules the CHR Way. *Fourth Workshop on Constraint Handling Rules*, pages 49–62, 2007.
8. Thom Frühwirt and Slim Abdennadher. *Essentials of Constraint Programming*. Springer-Verlag, 2003.
9. Yi Jin and Michael Thielscher. Representing beliefs in the fluent calculus. In *ECAI*, pages 823–827, 2004.
10. Antonis C. Kakas, Robert A. Kowalski, and Francesca Toni. Abductive logic programming. *Journal of Logic and Computation*, 2(6):719–770, 1992.
11. Maurice Pagnucco. The Role of Abductive Reasoning within the Process of Belief revision. Technical report, University of Sydney, 1996.
12. Raymond Reiter. *Readings in nonmonotonic reasoning*. Morgan Kaufmann Publishers Inc., 1980.
13. Michael Thielscher. *Reasoning Robots: The Art and Science of Programming Reasoning Agents*. Applied Logic Series 5. Kluwer, 2005.
14. Peter Van Weert, John Sneyers, Tom Schrijvers, and Bart Demoen. Extending CHR with Negation as Absence. *Third Workshop on Constraint Handling Rules*, pages 125–140, 2006.
15. Armin Wolf, Jacques Robin, and Jairson Vitorino. Adaptive CHR meets $CHR^{\vee}$: An Extended Refined Operational Semantics for $CHR^{\vee}$ based on Justifications. In *Proceedings of the Fourth Workshop on Constraint Handling Rules (CHR 2007)*, pages 1–15, Porto, Portugal, 2007.
16. Guizhen Yang and Michael Kifer. Inheritance in Rule-Based Frame Systems: Semantics and Inference. *Journal on Data Semantics (JoDS)*, II:79–135, 2006.