Description Logic and Rules the CHR Way
Extended Abstract

Thom Frühwirth

Faculty of Computer Science
University of Ulm, Germany
www.informatik.uni-ulm.de/pm/mitarbeiter/fruehwirth/

CHR 2007, Porto, September '07

Motivation

Semantic Web

Description Logic (DL) as logical knowledge representation language for ontologies, e.g. OWL.
Combine DL with rules for reasoning, e.g. SWRL, OWL-DL.
Rules are taken from logic programming in the broad sense.
But constraint-based approaches have not been considered so far.

Constraint Handling Rules (CHR)

Logical concurrent committed-choice guarded rules with built-in constraints.

Idea: Use CHR for DL reasoning.

Outline

- Constraint Handling Rules (CHR)
- Description Logic in CHR
- DL Rules in CHR

Constraint Handling Rules (CHR)

- Constraint programming language for Computational Logic
- Multi-headed guarded committed-choice rules
  transform multi-set of constraints until exhaustion
- Ideal for executable specifications and rapid prototyping
- Can implement algorithms with optimal time and space complexity
- Incrementality (on-line, any-time) and concurrency for free
- Logical and operational semantics coincide strongly
- High-level supports program analysis and transformation:
  Confluence/completion, operational equivalence, termination/time complexity, correctness...
- Implemenations in most Prolog systems, Java, Haskell
- 100s of applications from types, time tabling to cancer diagnosis

A CHR Constraint Solver for DL

A-Box and T-Box, Concepts and Roles

A-box (assertional knowledge):
conjunction of membership and role-filler assertions.
$a : s$ is a *membership assertion (constraint)*
$(a, b) : r$ is a *role-filler assertion (constraint)*,
where a and b are individuals (objects), s is a ground concept term,
and r is a role name.

T-box (terminological knowledge):
finite set of acyclic *concept definitions* c `isa` s,
where c is a concept name.

Implementing DL as Constraint System in CHR

$I : \texttt{not}\ S \leftrightarrow \neg(I : S)$	`I:not S, I:S <=> false` $(*)$
$I : S_1\ \texttt{and}\ S_2 \leftrightarrow I : S_1 \wedge I : S_2$	`I:S1 and S2 <=> I:S1, I:S2`
$I : S_1\ \texttt{or}\ S_2 \leftrightarrow I : S_1 \vee I : S_2$	`I:S1 or S2 <=> (I:S1 ; I:S2)`
$I : \texttt{some}\ R\ is\ S \leftrightarrow \exists J((I, J) : R \wedge J : S)$	`I:some R is S <=> (I,J):R, J:S`
$I : \texttt{all}\ R\ is\ S \leftrightarrow ((I, J) : R \rightarrow J : S)$	`I:all R is S, (I,J):R ==> J:S`
$C\ \texttt{isa}\ S \leftrightarrow (I : C \leftrightarrow I : S)$	`I:C <=> I:S,   I:not C <=> I:not`

$(*)$ Plus CHR rules to produce the Negation Normal Form.

Figure: FOL Constraint Theory and CHR Rules for $\mathcal{ALC}$

CHR Induced Properties

- Logical Correctness and Solved Normal Form
- Confluence
  Only clash rule overlaps with other rules.
- Termination
  Membership assertions in the body are strictly smaller than the ones in the head.
- Anytime and Online Algorithm Property
  We can stop the computation and restart it anytime while we get closer to the solved normal form and we can add assertions while the program runs without affecting correctness.
- Concurrency
  Each constraint can be handled in its own thread by and-parallelism (and or-parallelism for concept union). Sychronisation when the clash rule and the propagation rule for value restrictions are applied.

Complexity and Optimizations

All rules of our DL program can be applied in constant time, given an index on the first argument and role name of assertions.

Exponential complexity because of disjunction (and negation leading to it), multi-headed propagation rule and for value restriction.

Linear complexity in size of unfolded A-Box, polynomial in size of A-Box (after some optimizations).

Some CHR techniques to tame complexity:

- Enforce set-based semantics of assertions:
  `IJ:CR \ IJ:CR <=> true`
- Disjunction only if no other rule is applicable (labeling)
- Restrict applicability of expensive rules:
  `I:all R is D \ I:some R is S <=> (I,J):R, J:S`
  `X:C1 \ X:C <=> X:D`

DL techniques like cashing, blocking and trace technique can be implemented in CHR.

DL Extensions in CHR

Top (universal) and bottom (empty) concepts:

```
X:top <=> true.        X:bot <=> false.
```

Allsome quantifiers, e.g. `parent isa allsome child is human`:

```
I:allsome R is S <=> I:all R is S, I:some R is S
```

Role chains (nested roles), e.g. `grandfather isa father of father`:

```
(I,J):A of B <=> (I,K):A, (K,J):B
```

Inverse and Transitive Roles

```
(I,J):inv(R) ==> (J,I):R.     (I,J):R ==> (J,I):inv(R).
(I,K):R, (K,J):trans(R) ==> (I,J):trans(R)
```

Functional roles (features, attributes):

```
(I,J):F, (I,K):F ==> feature(F) | J=K.
```

Distinct, disjoint primitive concepts:

```
I:C1, I:C2 ==> distinct(C1), primitive(C2) | C1=C2.
```

Nominals (named individuals, singleton concepts) `X:{I} ==> X=I.`

Concrete domains (constraints from other domains):

```
(I,J):smaller ==> I<J.
```

Inclusion between concept terms, $C \sqsubseteq S$

```
I:C ==> I:S.     InotS ==> I:not C
```

DL Rules in CHR

In DL's, role-filler assertions only admit a tree structure.
E.g. cannot define `uncle` role as a male sibling of a person's father.

SWRL extends simple polynomial-time DL with material implication. SWRL is already undecidable.
Uncle example in SWRL translated to CHR:

```
male(Z), hassibling(Y,Z), hasparent(X,Y) ==> hasuncle(X,Z).
```

CHR performs bottom-up closure using propagation rules.

OWL-DL extends SWRL with non-DL atoms and disjunction:
$A_1 \vee \ldots \vee A_n \leftarrow B1 \wedge \ldots B_m$
Needs theorem prover.
Can still be implemented in CHR, starting from:

```
B1,...Bn ==> (A1 ; ...; An)
```

In the most general case, use clausal representation.

Conclusions

DL in CHR
Work in Progress

- Complete anytime and online algorithm for consistency checking of DL.
- Concise and compact set of rules with performance guarantees.
- Correct, confluent, and concurrent.
- Optimizations from constraint-programming and DL possible.
- DL Rules as CHR propagation rules, e.g. for SWRL and OWL-DL.
- In CHR, can integrate other constraint systems as concrete domains.
- In CHR, unbound variables (unsafe rules) pose no problem.

Future Work
Deepen understanding of relation between DL Rules and CHR.
Explore nonmonotonic aspects of DL Rules.