Meaning in Context
– constraint programming for natural language analysis

Henning Christiansen

Professor, PhD
Roskilde University, Computer Science
P.O.Box 260, DK-4000 Roskilde, Denmark
henning@ruc.dk, http://www.ruc.dk/ henning

Joint work with Veronica Dahl, Simon Fraser University, Canada

Work on constraints in Roskilde

- CONTROL project, 2004–2007, funded by Danish SNF
  "Constraints for Robust Languages processing"
  HC, John Gallagher, Jørgen Villadsen & assoc. Veronica Dahl, Ph. Blache
- Attention to Constraint Handling Rules
- CHR Grammars (HC, TPLP 2005). Grammar notation on top of CHR.
- Aspects of NLP, extensions to Prolog & CHR, abduction (HC,VD)
- Analysis and optimization of CHR (HC, JG)
- Looking for practical applications, e.g. deaf peoples' sign language
- CSLP workshop, Roskilde 2004. LNAI 3834; Barcelona 2005?
  Int'l workshop on Constraint Solving and Language Processing

Meaning in Context

An illustration of how CLP attitude may bring clarity into NLP.

- Simplified notion of meaning that fits better with pragmatics
- Basis for co-routining among different layers of analysis
- Each sentence interpreted and understood in context, and contributes to context

Techological background . . .

Techological background

- CHR, Constraint Handling Rules [Frühwirth, 1995, . . .]
  – declarative extension to Prolog for writing constraint solvers
  – available in SICStus Prolog (among others)

- A smart way of doing *abduction* in Logic Programming with CHR
  – efficient & light-weight
  – negation limited to so-called explicit negation

Constraint Handling Rules, intro. by example

```
:- use_module(library(chr)).
handler leq.
constraints leq/2.
:- op(500, xfx, leq).
X leq Y , Y leq Z ==> X leq Z.
X leq Y , Y leq X <=> X=Y.
X leq Y <=> X=Y | true.
X leq Y \ X leq Y <=> true.

p(X,Y):- q(X), r(Y,Z), X leq Z.
...
```

Execution model: Constraint store, replace/add constraints
Declarative semantics: as indicated by arrow symbols

Abduction in Logic Programming

Reasoning to find those “missing facts” of a Prolog program necessary to make given query succeed.

- Hot topic at logic programming conf. in 90ies
- Applications:
  – planning (event calculus and otherwise)
  – diagnosis
  – view updates in databases
- Typical implementations by meta-interp., $\approx$ 20–100 $\times$ slower that Prolog

An *abductive logic program* is a Prolog program extended with

- A set of distinguished, *abducible* predicates
- A set of *integrity constraints* to be satisfied by facts invented by interpreter

Abduction in Prolog + a little bit of CHR

Our trick:

- Abducibles $\rightarrow$ CHR constraints
- Integrity constraints $\rightarrow$ CHR rules
- Abductive programs run as Prolog, but with CHR taking care of abducibles

Example, view update in a database...

Abduction in Prolog + a little bit of CHR

Example, view update in a database

```
constraints father/2, mother/2, male/1, female/2.
parent(X,Y):- father(X,Y) ; mother(X,Y).
initial_db:- father(peter,john), male(peter).

mother(X,_) ==> female(X).
father(X,Z),father(Y,Z) ==> X=Y.
...

?- initial_db, parent(jane,john).
...
mother(jane,john),
female(jane) ?
```

Analysis of (natural) language as abduction

Given discourse assumed faithful to some "real world"
Context: (Partial) knowledge about this world

Basic assumption: Grammar $\wedge$ Context $\rightarrow$ Sentences

Observe: This works fine with Prolog (DCG) to generate or verify given discourse *known* Context.

Discourse *analysis* is an abductive problem.

An example. . .

An example, analysis of discourses about still-lifes

Integrity constraints about abducible context facts:

```
i_on(X,Y), i_on(Y,X) ==> fail.     container(C) ==> thing(C).
i_in(the_box,the_vase) ==> fail.  i_in(_,C) ==> container(C).
...
thing(X) ==> X=the_flower ; X=the_box ; X=the_vase ; X=the_table.
container(X) ==> X=the_box ; X=the_vase.
container(the_flower) ==> fail. container(the_table) ==> fail.
on(X,Y) ==> i_on(X,Y) ; i_on(X,Z), i_on(Z,Y) ; i_in(X,Z), i_on(Z,Y).
```

The grammar:

```
sentence --> [A,is,on,B], {thing(A), thing(B), on(A,B)}.
```

Query: `?- phrase(sentence, [the,flower,is,on,the,table]`
One of the answers: `i_in(the_flower,the_vase), i_on(the_vase,the_table)`,
...

Meaning-in-context: Taking pragmatics serious

He won it

The tall, red-haired man carrying a laptop won a brand new Ferrari

Both may have the same meaning or purpose to express: `won(X,Z)`
where `X` and `Y` are references to objects in (presupposed) context:

`tall(X), read_haired(X), carries(X,Y), laptop(Y), ferrari(Z), brand_new(Z)`

Or to model a passive agent spying a discourse:

Sentence meaning: $\emptyset$

Sentence presupposes context:
`won(X,Z)`, `tall(X)`, ...

As opposed to std. Montague semantics with context-independent λ-terms ...

Perspectives

- Separating out context greatly simplifies semantic terms
- Seems more "pragmatics-oriented"
- Allows all levels of analysis interact with context,
  – resolve lexical ambiguities
  – identify and employ predefined contexts,
  `gear-box` ... `brakes` ... `200kmh` ... `==> predef_context(all_about_cars)`
- Approach is formalized by *possible-worlds semantics* [not shown today]; formalization and generalization of R. Stalnaker's theories about context and accommodation

Conclusion

- Abduction in Logic Programming can be implemented elegantly and efficiently in CHR
- Works with CHR Grammars [HC, 2001, ..., TPLP 2005] and Prolog in A^2LP paradigm [HC,VD, MultiCPL 2004]
- Constraint programming (CHR) revives abduction for NL interpretation
- ... and gives inspiration to Meaning-in-Context model