Compiling Input$*$FO($\cdot$) Inductive Definitions into Tabled Prolog Rules for IDP3

Joachim Jansen,
Gerda Janssens,
Albert Jorissen

KU LEUVEN

DTAI
Declaratieve Talen en Artificiële Intelligentie

Take away message

- IDP : Knowledge Base System (ASP-like)
- Intentional predicates + large datasets
  - Tried semi-naive approach :-(
- Use Tabled Prolog in IDP

Outline

- Introduction
- Transformations
- Implementation
- Experiments
- Future work

Outline

- Introduction
- Transformations
- Implementation
- Experiments
- Future work

IDP3

- Knowledge Base System
- Modeling language FO(·)
- Ground (top-down) + Solve (MinisatID)

Introduction : NQueens in IDP

$$
\begin{array}{l}
vocabulary\ \ V\ \{ \\
\quad type\ index\ isa\ \text{int} \\
\quad type\ diag\ isa\ \text{int} \\
\quad queen(index, index) \\
\quad n\ :\ index \\
\quad diag1(index, index)\ :\ diag \\
\quad diag2(index, index)\ :\ diag \\
\}
\end{array}
$$

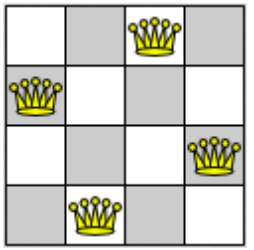
$$
\begin{array}{l}
theory\ T\ :\ V\ \{ \\
\quad \forall x[index]\ :\ \exists_{=1} y\ :\ queen(x, y). \\
\quad \forall y[index]\ :\ \exists_{=1} x\ :\ queen(x, y). \\
\quad \forall d[diag]\ :\ \#\{\ x[index], y[index]\ :\ queen(x, y) \wedge diag1(x, y) = d\ \} < 2. \\
\quad \forall d[diag]\ :\ \#\{\ x[index], y[index]\ :\ queen(x, y) \wedge diag2(x, y) = d\ \} < 2. \\
\quad \{\ \forall x[index] y[index]\ :\ diag1(x, y) = x - y + n.\ \} \\
\quad \{\ \forall x[index] y[index]\ :\ diag2(x, y) = x + y - 1.\ \} \\
\}
\end{array}
$$

Introduction : NQueens in IDP

```
structure S : V {
    n = 4
    index = {1..4}
    diag = {1..7}
    queen = ?
}
```

```
procedure main {
    sol = modelexpand(T, S)[1]
    print(sol)
}
```

```
structure S : V {
    n = 4
    index = {1..4}
    diag = {1..7}
    queen = {1,2 ; 2,4 ; 3,1 ; 4,3}
}
```



Introduction : Input$*$ predicates

$$\{\ \forall x[index]\, y[index]\ :\ diag1(x, y) = x - y + n.\ \}$$

$$\{\ \forall x\, y : reaches(x, y)\ \leftarrow\ edge(x, y).$$
$$\forall x\, y : reaches(x, y)\ \leftarrow\ reaches(x, z)\ \wedge\ reaches(z, y).\ \}$$

Input$*$ predicates depend only on

- built-ins (e.g. +/2)
- part of input (e.g. edge/2, n/0)
- other Input$*$ predicates

Calculate before grounding

Introduction – Going from IDP to Tabled Prolog

Language

- (Partial) functions
  - ≠ “functions” in LP (constructors)
- (Sub)types
- Aggregates
- Quantifiers

Execution

- Ground + solve (SAT search)

  vs. SLD resolution + Tabling

Outline

- Introduction
- Transformations
- Implementation
- Experiments
- Future work

Transformations

- Two components to translate

  Theory + Structure

- Structure: transform into facts + ranges

edge = {a,c; c,b; a,c}

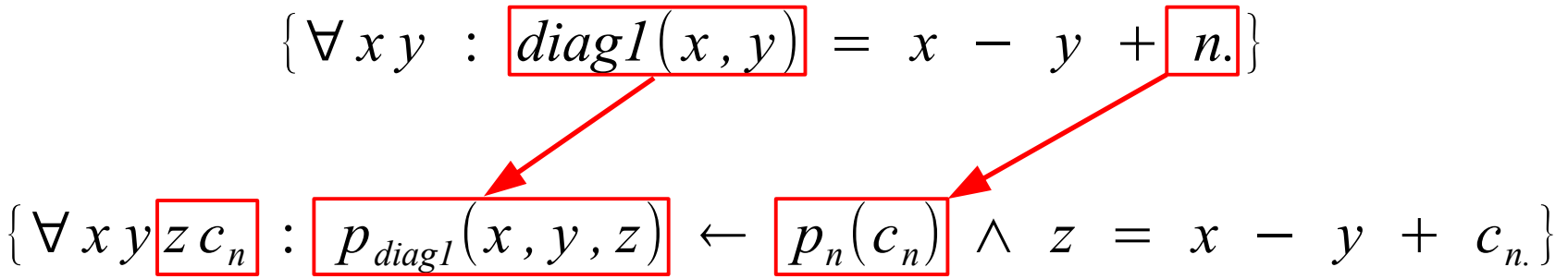
```
edge(a,b).
edge(c,b).
edge(a,c).
```

index = {1..4}

```
index(X) :- between(1,4,X).
```

Transformations - Theory

- Pre-transformations
  - Function Elimination



  - Implication Elimination, Equivalence Elimination, Flattening, Negation Pushing, Tseitinisation

Transformations - Theory

- 4 rule normalforms

typed

possibly negated

$$\forall \boxed{\bar{x}_1 \ldots \bar{x}_n} \; : \; p(\bar{x}_1 \ldots \bar{x}_n) \leftarrow \boxed{p_1(\bar{x}_1)} \wedge \boxed{p_2(\bar{x}_2)} \wedge \ldots \wedge \boxed{p_n(\bar{x}_n)}.$$

$$\forall \bar{x}_1 \ldots \bar{x}_n \; : \; p(\bar{x}_1 \ldots \bar{x}_n) \leftarrow p_1(\bar{x}_1) \vee p_2(\bar{x}_2) \vee \ldots \vee p_n(\bar{x}_n).$$

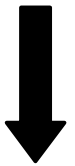
$$\forall \bar{x}_1 \; : \; p(\bar{x}_1) \leftarrow \exists \bar{x}_2 \; : \; p_e(\bar{x}_1, \bar{x}_2).$$

$$\forall \bar{x}_1 \; : \; p(\bar{x}_1) \leftarrow \forall \bar{x}_2 \; : \; p_f(\bar{x}_1, \bar{x}_2).$$

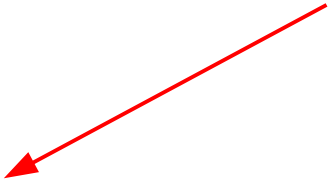
Transformations – Unsafe Variables

- Prolog variables that
  - do not occur in a positive goal in the body
  - occur as input arguments for Prolog built-ins

```
p(X,Y,Z) :- not(q(Y)), X < Y.
```



```
p(X,Y,Z) :-
     index(Y), not(q(Y)),
     index(X), X < Y,
     type_z(Z).
```



"typesunsafe"

```
index(X) :- between(1,4,X).
```

Transformations - Existential

$$\forall \bar{x}_1 \ : \ p(\bar{x}_1) \leftarrow \exists \bar{x}_2 \ : \ p_e(\bar{x}_{1,} \bar{x}_2).$$

```
p(X1) :- typesunsafe, pe(X1,X2).
```

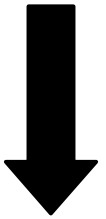
Transformations - Disjunction

$$\forall \bar{x}_1 \ldots \bar{x}_n \ : \ p(\bar{x}_1 \ldots \bar{x}_n) \leftarrow \ p_1(\bar{x}_1) \ \vee \ p_2(\bar{x}_2) \ \vee \ \ldots \ \vee \ p_n(\bar{x}_n).$$

```
p(X) :- typesunsafe, p1(x1).
p(X) :- typesunsafe, p2(x2).
...
p(X) :- typesunsafe, pn(xn).
```

Transformations - Universal

$$\forall \bar{x}_1 \; : \; p(\bar{x}_1) \leftarrow \forall \bar{x}_2 \; : \; p_f(\bar{x}_1, \bar{x}_2).$$



```
p(X1) :-
    type_X1(X1),
    idp_forall(type_X2(X2),p_f(X1,X2)).
```

$$(type_{X2} \Rightarrow p_f) \quad \Leftrightarrow \quad \neg(type_{X2} \text{ and } \neg p_f)$$

```
idp_forall(C1,C2) :-
    tables:not_exists((call(C1),
        tables:not_exists(C2))
    ).
```

Transformations - Conjunction

$$\forall \bar{x}_1 \ldots \bar{x}_n \; : \; p(\bar{x}_1 \ldots \bar{x}_n) \leftarrow p_1(\bar{x}_1) \wedge p_2(\bar{x}_2) \wedge \ldots \wedge p_n(\bar{x}_n).$$

```
p(X) :- typesunsafe, p1(x1), p2(x2) ... pn(xn).
```

Outline

- Introduction
- Transformations
- Implementation
- Experiments
- Future work

Prolog Workflow

- Compile transformed rules
- Dynamically assert facts
- Query predicates with variables to collect all answers
- Export answers to IDP and remove definition of Input$*$ predicates

Implementation

- XSB
  - Tabling (recursion)
  - Well-Founded Semantics
  - Well-documented C interface
- Settings: local scheduling, variant tabling

Outline

- Introduction
- Transformations
- Implementation
- Experiments
- Future work

Experiments

Problem	# input*	Size	IDP3		IDP3 XSB		CLINGO		DLV
			ground	solve	ground	solve	ground	solve	solve
REACH	2/2	10	0.06	0.06	0.09	0.09	0.01	0.01	0.01
		40	0.24	0.24	0.13	0.13	0.01	0.01	0.02
		400	106.04	106.08	1.50	1.50	0.27	0.44	4.09
PATH10	1/1	50	0.18	0.24	0.09	0.09	0.01	0.01	0.01
		300	6.00	6.00	0.25	0.25	0.01	0.01	0.04
HP	0/3	40	0.09	0.09	0.09	0.09	0.01	0.01	0.05
		400	4.15	5.00	4.15	5.00	0.03	0.15	9.34
NQ	2/3	5	0.18	0.18	0.09	0.09	0.01	0.01	0.01
		20	0.78	0.78	0.13	0.13	0.01	0.01	0.09
		100	226.07	229.09	0.39	0.58	0.10	0.20	-
HNQ	4/5	5	0.32	0.33	0.15	0.16	0.01	0.02	0.02
		20	1.52	1.53	0.80	0.18	0.02	0.03	0.23
		50	39.80	40.10	24.80	0.79	0.08	0.17	-
		100	-	-	-	4.08	0.36	0.64	-

Available at http://dtai.cs.kuleuven.be/krr/research/experiments

Outline

- Introduction
- Transformations
- Implementation
- Experiments
- Future work

Future work

- Reduce overhead
  - Lower level integration with XSB
- Subsumptive tabling
- Relax conditions on Input$*$ predicates

Take away message

- IDP : Knowledge Base System (ASP-like)
- Intentional predicates + large datasets
  - Tried semi-naive approach :-(
  - XSB port :-)
- Contribution: Translate FO(·) definitions to Tabled Prolog + efficiently calculate them

End

Thank you.