CHRiSM: Chance Rules induce Statistical Models

Jon Sneyers, Wannes Meert and Joost Vennekens
K.U.Leuven, Belgium

CHR’09 Workshop
Pasadena, California, USA, July 2009
CHRiSM: Chance Rules induce Statistical Models

1 Introduction
   - Constraint Handling Rules
   - PCHR

2 CHRiSM

3 Implementation of CHRiSM

4 Discussion and conclusion
High-level language *extension*

- different host languages (originally and mostly Prolog)
- e.g. CHR(Prolog), CHR(Haskell), CHR(Java), CHR(C)

Multi-headed committed-choice guarded rewrite rules

Originally: designed for writing constraint solvers

Today: general-purpose programming language
Semantics of CHR

- **Declarative semantics:**
  - classical logic semantics
  - linear logic semantics [Betz & Frühwirth 2005]

- **Abstract operational semantics** $\omega_t$

- **Refined operational semantics** $\omega_r$ [Duck et al. 2004]
  - activate constraints depth-first, left-to-right
  - search for matching rules by trying occurrences in textual order

- **Priority semantics** $\omega_p$ [De Koninck et al. 2007]
  - apply rules in order of priority
  - also dynamic priorities
- Probabilistic CHR: rules get a weight
- Coin toss in PCHR:

**PCHR program**

```
toss <= 0.5: head.
toss <= 0.5: tail.
```

- Semantics: $\omega_t$ semantics, where the applied rule is probabilistically chosen from all applicable rules
- Probability distribution given by rule weights: probability = normalized weight
Problems with PCHR

- Only simplification rules
- Not easy to see probability that a rule is applied:
  - depends on its weight and weights of all other applicable rules
  - meaning of a rule is non-local: depends on the entire program
- PCHR instantiates $\omega_t$, but its semantics depends on the full non-determinism of $\omega_t$
  - ‘refined’ or ‘priority’ PCHR is probably not desirable
  - efficient implementation of PCHR is not straightforward
Introduction

2 CHRiSM
- Syntax & semantics
- Examples
- PRISM features in CHRiSM

3 Implementation of CHRiSM

4 Discussion and conclusion

CHRiSM: Chance Rules induce Statistical Models
New proposal for probabilistic CHR: CHRiSM

based on CHR(PRISM)

PRISM: PRogramming In Statistical Modeling
[Sato 1995, Sato & Kameya 1997]

CHRiSM: CHance Rules induce Statistical Models
- Chance rules (may) have two kinds of probabilities:
  - Rule: application probability
  - Body: probabilistic disjunction

Syntax: rule with probability Prob

```
Prob ?? Head <=> Guard | Body.
```

default: “1 ??” (normal CHR rule)

Syntax: probabilistic disjunction (in rule body)

fixed probability distribution: (cf. CP-Logic [Vennekens et al. 2006])

```
D1:Prob1 ; D2:Prob2 ; ... ; DN:ProbN
```

unknown probability distribution:

```
Prob ?? D1 ; D2 ; ... ; DN
```
Chance rules (may) have two kinds of probabilities:
- Rule: application probability
- Body: probabilistic disjunction

Syntax: rule with probability \( \text{Prob} \)

\[
\text{Prob} \ ?\ \text{Head} \iff \text{Guard} \mid \text{Body}.
\]

Default: “1 ??” (normal CHR rule)

Syntax: probabilistic disjunction (in rule body)

Fixed probability distribution: (cf. CP-Logic [Vennekens et al. 2006])

\[
\text{D1:Prob1} \ ; \ \text{D2:Prob2} \ ; \ldots \ ; \ \text{DN:ProbN}
\]

Unknown probability distribution:

\[
\text{Prob} \ ?\ \text{D1} \ ; \ \text{D2} \ ; \ldots \ ; \ \text{DN}
\]
Syntax of chance rules

- Chance rules (may) have two kinds of probabilities:
  - Rule: application probability
  - Body: probabilistic disjunction

Syntax: rule with probability Prob

\[
\text{Prob} \text{ ?? Head} \leftrightarrow \text{Guard} \mid \text{Body}.
\]

Default: “1 ??” (normal CHR rule)

Syntax: probabilistic disjunction (in rule body)

Fixed probability distribution: (cf. CP-Logic [Vennekens et al. 2006])

\[
D1:\text{Prob}1 \; ; \; D2:\text{Prob}2 \; ; \; \ldots \; ; \; DN:\text{Prob}N
\]

Unknown probability distribution:

\[
\text{Prob} \; ?? \; D1 \; ; \; D2 \; ; \; \ldots \; ; \; DN
\]
Different kinds of probability expressions \textit{Prob} allowed:

- numbers:
  \texttt{head:0.5 ; tail:0.5}

- arithmetic expression which is ground at runtime:
  \texttt{eval(1-K) ?? maybe\_keep\_me(K) \iff true.}

- probabilities are unknown:
  \texttt{roll \iff 1 ; 2 ; 3 ; 4 ; 5 ; 6}

- probabilities are unknown and parametrized:
  \texttt{roll(\texttt{Die}) \iff Die \iff 1 ; 2 ; 3 ; 4 ; 5 ; 6}

Numbers and arithmetic expressions: fixed probabilities

Parametrized probabilities (with 0 or more parameters):

- initially: uniform distribution
- actual distribution can be learned from examples
Operational semantics as usual \((\omega_t, \omega_r, \omega_p)\)

- Two differences:
  - rule application can be skipped (with probability \(1 - P\))
  - probabilistic disjunctions in the body: one disjunct is randomly chosen (committed-choice)
Example 1: coin toss

CHRiSM program

```
toss <=> head:0.5 ; tail:0.5.
```

Example interaction

```
?- sample toss
toss <==> tail.
```
Example 1: coin toss

CHRiSM program

\texttt{toss} \leftrightarrow \texttt{?? head;tail}.

Example interaction

\begin{verbatim}
?- sample toss
\texttt{toss} \leftrightarrow \texttt{tail}.

?- sample toss,toss
\texttt{toss} \leftrightarrow \texttt{head;tail}.
\end{verbatim}

Probability of \texttt{toss,toss} \leftrightarrow \texttt{head,head} is 0.25
Example 1: coin toss

**CHRiSM program**

toss <=> ?? head;tail.

**Example interaction**

```
| ?- sample toss
  toss <=> tail.
| ?- sample toss,toss
  toss,toss <=> head,head.
| ?- prob toss,toss <=> head,head
  Probability of toss,toss<==>head,head is: 0.25
```
Example 1: coin toss

CHRiSM program

\[ \text{toss} \leftrightarrow \text{?? head;tail}. \]

Example interaction

\[
| \text{?- sample toss} \\
\text{toss} \leftrightarrow \text{tail}. \\
| \text{?- sample toss,toss} \\
\text{toss,toss} \leftrightarrow \text{head,tail}. \\
| \text{?- prob toss,toss} \leftrightarrow \text{head,head} \\
\text{Probability of toss,toss} \leftrightarrow \text{head,head is: 0.25}
\]
**Example 1: coin toss**

**CHRiSM program**

```
toss <=> ?? head;tail.
```

**Example interaction**

```
| ?- sample toss
| toss <=> tail.
| ?- sample toss,toss
| toss,toss <=> head,tail.
| ?- prob toss,toss <=> head,head
Probability of toss,toss <=> head,head is: 0.25
```
Example 1: coin toss

CHRiSM program

\begin{verbatim}
toss <=> ?? head;tail.
\end{verbatim}

Example interaction

\begin{verbatim}
| ?- sample toss
| toss <=> tail.
| ?- sample toss,toss
| toss,toss <=> head,tail.
| ?- prob toss,toss <=> head,head
Probability of toss,toss<=>head,head is: 0.25
\end{verbatim}
Example 1: coin toss

CHRiSM program

\[
toss \iff ?? \text{head;tail}.
\]

Example interaction

\[
| \ ?- \ \text{sample toss}
toss \iff \text{tail}.
| \ ?- \ \text{sample toss,toss}
toss,toss \iff \text{head, tail}.
| \ ?- \ \text{prob toss,toss} \iff \text{head,head}
\]

Probability of toss,toss\iff\text{head,head} is: 0.25
Example 1: coin toss

CHRiSM program

toss <-> ?? head;tail.

Example interaction

| ?- sample toss
| toss <-> tail.
| ?- sample toss,toss
| toss,toss <-> head,tail.
| ?- prob toss,toss <-> head,head
Probability of toss,toss<==>head,head is: 0.25
**Example 2: rock-scissors-paper**

**CHRiSM program**

```
player(P) <=> P ?? rock(P) ; scissors(P) ; paper(P).

rock(P1), scissors(P2) ==> winner(P1).
scissors(P1), paper(P2) ==> winner(P1).
paper(P1), rock(P2) ==> winner(P1).
```

**Example interaction**

```
| ?- sample player(tom),player(jon)
player(tom),player(jon) <=> rock(jon),rock(tom).
| ?- sample player(tom),player(jon)
player(tom),player(jon) <=> rock(jon),paper(tom).
player(tom),player(jon) <=> scissors(jon),rock(tom).
player(tom),player(jon) <=> scissors(jon),paper(tom).
player(tom),player(jon) <=> paper(jon),rock(tom).
player(tom),player(jon) <=> paper(jon),scissors(tom).
player(tom),player(jon) <=> scissors(tom),rock(tom).
player(tom),player(jon) <=> scissors(tom),paper(tom).
```

Probability of player(tom),player(jon) ==> winner(tom) is: 0.33333
Example 2: rock-scissors-paper

**CHRiSM program**

```prolog
player(P) <=> P ?? rock(P) ; scissors(P) ; paper(P).

rock(P1), scissors(P2) ==> winner(P1).
scissors(P1), paper(P2) ==> winner(P1).
paper(P1), rock(P2) ==> winner(P1).
```

**Example interaction**

```prolog
| ?- sample player(tom),player(jon)
player(tom),player(jon) <=> rock(jon),rock(tom).
| ?- sample player(tom),player(jon)
player(tom),player(jon) <=> rock(jon),paper(tom),winner(tom).
| ?- prob player(tom),player(jon) ===> winner(tom)
Probability of player(tom),player(jon) ===> winner(tom) is: 0.33333
```
Example 2: rock-scissors-paper

**CHRiSM program**

```prolog
player(P) <=> P ?? rock(P) ; scissors(P) ; paper(P).

rock(P1), scissors(P2) ==> winner(P1).
scissors(P1), paper(P2) ==> winner(P1).
paper(P1), rock(P2) ==> winner(P1).
```

**Example interaction**

```
| ?- sample player(tom),player(jon) player(tom),player(jon) <=> rock(jon),rock(tom).
| ?- sample player(tom),player(jon) player(tom),player(jon) <=> rock(jon),paper(tom),winner(tom).
| ?- prob player(tom),player(jon) ==> winner(tom) Probability of player(tom),player(jon)===>winner(tom) is: 0.33333
```
Example 2: rock-scissors-paper

**CHRiSM program**

```plaintext
player(P) <=> P ?? rock(P) ; scissors(P) ; paper(P).

rock(P1), scissors(P2) ==> winner(P1).
scissors(P1), paper(P2) ==> winner(P1).
paper(P1), rock(P2) ==> winner(P1).
```

**Example interaction**

```
| ?- sample player(tom), player(jon)
player(tom), player(jon) <=> rock(jon), rock(tom).
| ?- sample player(tom), player(jon)
player(tom), player(jon) <=> rock(jon), paper(tom), winner(tom).
| ?- prob player(tom), player(jon) ==> winner(tom)
Probability of player(tom), player(jon) ==> winner(tom) is: 0.3333
```
**Example 2: rock-scissors-paper**

**CHRiSM program**

```prolog
player(P) <=> P ?? rock(P) ; scissors(P) ; paper(P).

rock(P1), scissors(P2) ==> winner(P1).
scissors(P1), paper(P2) ==> winner(P1).
paper(P1), rock(P2) ==> winner(P1).
```

**Example interaction**

```
| ?- sample player(tom), player(jon)
player(tom), player(jon) <=> rock(jon), rock(tom).
| ?- sample player(tom), player(jon)
player(tom), player(jon) <=> rock(jon), paper(tom), winner(tom).
| ?- prob player(tom), player(jon) ==> winner(tom)
Probability of player(tom), player(jon) ==> winner(tom) is: 0.33333
```
Example 2: rock-scissors-paper

CHRiSM program

\[
\text{player}(P) \implies P \bowtie \text{rock}(P) \bowtie \text{scissors}(P) \bowtie \text{paper}(P).
\]

\[
\text{rock}(P1), \text{scissors}(P2) \implies \text{winner}(P1).
\]

\[
\text{scissors}(P1), \text{paper}(P2) \implies \text{winner}(P1).
\]

\[
\text{paper}(P1), \text{rock}(P2) \implies \text{winner}(P1).
\]

Example interaction

\[
\text{?- sample player(tom),player(jon)}
\]

\[
\text{player(tom)}, \text{player(jon)} \implies \text{rock(jon)}, \text{rock(tom)}.
\]

\[
\text{?- sample player(tom),player(jon)}
\]

\[
\text{player(tom)}, \text{player(jon)} \implies \text{rock(jon)}, \text{paper(tom)}, \text{winner(tom)}.
\]

\[
\text{?- prob player(tom),player(jon) \implies winner(tom)}
\]

Probability of player(tom), player(jon) \implies winner(tom) is: 0.33333
Example 2: rock-scissors-paper

**CHRiSM program**

```
player(P) <=> P ? rock(P) ; scissors(P) ; paper(P).

rock(P1), scissors(P2) ==> winner(P1).
scissors(P1), paper(P2) ==> winner(P1).
paper(P1), rock(P2) ==> winner(P1).
```

**Example interaction**

```
| ?- sample player(tom),player(jon)        
player(tom),player(jon) <=> rock(jon),rock(tom).
| ?- sample player(tom),player(jon)        
player(tom),player(jon) <=> rock(jon),paper(tom),winner(tom).
| ?- prob player(tom),player(jon) ==> winner(tom) 
Probability of player(tom),player(jon) ==> winner(tom) is: 0.33333
```
Example 3: generating random graphs

**CHRiSM program**

```
0.5 ?? node(A), node(B) ==> edge(A,B).
```

**Example interaction**

```
| ?- sample node(a),node(b),node(c)
node(a),node(b),node(c) <==>
node(c),node(b),node(a),edge(b,c),edge(c,b),edge(b,a).
| ?- sample node(a),node(b),node(c)
node(a),node(b),node(c) <==>
node(b),node(c),node(a),edge(c,b),edge(b,c),edge(c,a),
edge(b,a).
```
Example 3: generating random graphs

CHRiSM program

```plaintext
0.5 ?? node(A), node(B) ==> edge(A,B).
```

Example interaction

```plaintext
| ?- sample node(a),node(b),node(c)
node(a),node(b),node(c) <==>
node(c),node(b),node(a),edge(b,c),edge(c,b),edge(b,a).  
| ?- sample node(a),node(b),node(c)
node(a),node(b),node(c) <==>
node(c),node(b),node(a),edge(a,c),edge(b,c),edge(c,a),
  edge(c,b),edge(a,b).
```
Example 3: generating random graphs

**CHRiSM program**

0.5 ?? node(A), node(B) ==> edge(A,B).

**Example interaction**

```
| ?- sample node(a),node(b),node(c)
node(a),node(b),node(c) <=>
node(c),node(b),node(a),edge(b,c),edge(c,b),edge(b,a).
| ?- sample node(a),node(b),node(c)
node(a),node(b),node(c) <=>
node(c),node(b),node(a),edge(a,c),edge(b,c),edge(c,a),
    edge(c,b),edge(a,b).
```
Example 3: generating random graphs

CHiRSM program

0.5 ?? node(A), node(B) ==> edge(A,B).

Example interaction

| ?- sample node(a),node(b),node(c)
node(a),node(b),node(c) <==>
node(c),node(b),node(a),edge(b,c),edge(c,b),edge(b,a).
| ?- sample node(a),node(b),node(c)
node(a),node(b),node(c) <==>
node(c),node(b),node(a),edge(a,c),edge(b,c),edge(c,a),
edge(c,b),edge(a,b).
Example 3: generating random graphs

CHRiSM program

0.5 ?? node(A), node(B) ==> edge(A,B).

Example interaction

| ?- sample node(a), node(b), node(c)  
  node(a), node(b), node(c) <==>
  node(c), node(b), node(a), edge(b,c), edge(c,b), edge(b,a).
| ?- sample node(a), node(b), node(c)  
  node(a), node(b), node(c) <==>
  node(c), node(b), node(a), edge(a,c), edge(b,c), edge(c,a),
  edge(c,b), edge(a,b).
CHRiSM program

```
go => burglary(yes);burglary(no).
go => earthquake(yes);earthquake(no).
burglary(B), earthquake(E) =>
    B,E ?? alarm(yes);alarm(no).
alarm(A) => A ?? johncalls(yes);johncalls(no).
alarm(A) => A ?? marycalls(yes);marycalls(no).
```
PRISM has many nice features, a.o.:
- Probabilistic execution (sample)
- Probability computation (prob)
- EM-learning (learn)

These features can also be used in CHRIISM
- **Probabilistic execution:** sample **goal**
  - starting from **goal**, apply CHriSM rules (just like in CHR)
  - rules with probability $P$ are skipped with probability $1 - P$
  - in a probabilistic disjunction, exactly one disjunct is chosen

- **Probability computation:** $\text{prob } \text{goal} \iff \text{result}$
  - compute probability that "sample **goal" gives "**result"
  - $\text{prob } \text{goal} \implies \text{result}$
    - compute probability that "sample **goal" gives something of the form "**result, otherstuff"

- **EM-learning:** $\text{learn} (\text{observations})$
  - **observations**: a list of observations of the form
    "**goal \iff \text{result}" or "**goal \implies \text{result}"
  - compute an assignment to the unknown probabilities such that the likelihood of the observations is maximized
**Example: Learning**

**CHRiSM program**

```
player(P) <=> P ?? rock(P) ; scissors(P) ; paper(P).
rock(P1), scissors(P2) ==> winner(P1).
...```

**Example interaction**

```
| ?- learn([ count((player(tom),player(jon) ==> winner(tom)),50),
            count((player(tom),player(jon) ==> winner(jon)),20),
            count((player(tom),player(jon) ==> ~winner(tom),~winner(jon)),30)])
  ...
| ?- show_sw
Switch exp1(jon): 1 (p: 0.60057034) 2 (p: 0.06536821) 3 (p: 0.33406143)
Switch exp1(tom): 1 (p: 0.08420895) 2 (p: 0.20973622) 3 (p: 0.70605482)
| ?- prob player(tom),player(jon) ==> winner(tom)
Probability of player(tom),player(jon)==>winner(tom) is: 0.499604
```
### CHRiSM program

```
player(P) <=> P ?? rock(P) ; scissors(P) ; paper(P).
rock(P1), scissors(P2) ==> winner(P1).
...```

### Example interaction

```
?- learn([ count((player(tom),player(jon) ==> winner(tom)),50),
            count((player(tom),player(jon) ==> winner(jon)),20),
            count((player(tom),player(jon) ==> ~winner(tom),~winner(jon)),30)])
#goals: 0(3)
Exporting switch information to the EM routine ... done
#em-iterations: 0..(23) (Converged: -102.965335828)
Statistics on learning:
Graph size: 72
Number of switches: 2
Number of switch instances: 6
Number of iterations: 23
Final log likelihood: -102.965335828
Total learning time: 0.000 seconds
Explanation search time: 0.000 seconds
Total table space used: 40496 bytes
Type show_sw or show_sw_b to show the probability distributions....
?- show_sw
Switch expr(jon): 1 (p: 0.60057034) 2 (p: 0.06536821) 3 (p: 0.33406143)```
CHRiSM program

```prolog
player(P) <-> P ?= rock(P) ; scissors(P) ; paper(P).
rock(P1), scissors(P2) ==> winner(P1).
...```

Example interaction

```
?- learn([ count(player(tom),player(jon) ==> winner(tom)), 50),
           count(player(tom),player(jon) ==> winner(jon)), 20),
           count(player(tom),player(jon) ==> ~winner(tom), ~winner(jon)), 30])
...
?- show_sw
Switch exp1(jon): 1 (p: 0.60057034) 2 (p: 0.06536821) 3 (p: 0.33406143)
Switch exp1(tom): 1 (p: 0.08420895) 2 (p: 0.20973622) 3 (p: 0.70605482)
?- prob player(tom),player(jon) ==> winner(tom)
Probability of player(tom),player(jon)==>winner(tom) is: 0.499604
```
**CHRiSM program**

```
player(P) <=> P ?? rock(P) ; scissors(P) ; paper(P).
rock(P1), scissors(P2) ==> winner(P1).
...
```

**Example interaction**

```
| ?- learn([ count((player(tom),player(jon) ===> winner(tom)),50),
           count((player(tom),player(jon) ===> winner(jon)),20),
           count((player(tom),player(jon) ===> ~winner(tom),~winner(jon)),30)])
| ...
| ?- show_sw
Switch exp1(jon): 1 (p: 0.60057034) 2 (p: 0.06536821) 3 (p: 0.33406143)
Switch exp1(tom): 1 (p: 0.08420895) 2 (p: 0.20973622) 3 (p: 0.70605482)
| ?- prob player(tom),player(jon) ===> winner(tom)
Probability of player(tom),player(jon) ===> winner(tom) is: 0.499604
```
CHiRM program

\[ \text{player(P)} \iff P \, ?\!\? \, \text{rock(P)} \, ; \, \text{scissors(P)} \, ; \, \text{paper(P)}. \]
\[ \text{rock(P1), scissors(P2)} \implies \text{winner(P1)}. \]

Example interaction

\[ \text{?- learn([ count((player(tom), player(jon)) \implies \text{winner(tom)}), 50),}
\]
\[ \text{count((player(tom), player(jon)) \implies \text{winner(jon)}), 20),}
\]
\[ \text{count((player(tom), player(jon)) \implies \sim \text{winner(tom)}, \sim \text{winner(jon)}), 30])}) \]
\[ \text{...} \]
\[ \text{?- show_sw} \]
\[ \text{Switch exp1(jon): 1 (p: 0.60057034) 2 (p: 0.06536821) 3 (p: 0.33406143)} \]
\[ \text{Switch exp1(tom): 1 (p: 0.08420895) 2 (p: 0.20973622) 3 (p: 0.70605482)} \]
\[ \text{?- prob player(tom), player(jon) \implies \text{winner(tom)}} \]
\[ \text{Probability of player(tom), player(jon) \implies \text{winner(tom)} is: 0.499604} \]
**Example: learning**

**CHRiSM program**

```
player(P) <=> P ?? rock(P) ; scissors(P) ; paper(P).
rock(P1), scissors(P2) ==> winner(P1).
...
```

**Example interaction**

```
| ?- learn([ count((player(tom),player(jon) ==> winner(tom)),50),
             count((player(tom),player(jon) ==> winner(jon)),20),
             count((player(tom),player(jon) ==> ~winner(tom),~winner(jon)),30)])
| ?- show_sw
Switch exp1(jon): 1 (p: 0.60057034) 2 (p: 0.06536821) 3 (p: 0.33406143)
Switch exp1(tom): 1 (p: 0.08420895) 2 (p: 0.20973622) 3 (p: 0.70605482)
| ?- prob player(tom),player(jon) ==> winner(tom)
Probability of player(tom),player(jon)===>winner(tom) is: 0.499604
```
**Example: learning**

**CHRiSM program**

```
player(P) <=> P ?? rock(P) ; scissors(P) ; paper(P).
rock(P1), scissors(P2) ==> winner(P1).
...
```

**Example interaction**

```
| ?- learn([ count((player(tom),player(jon) ==> winner(tom)),50),
             count((player(tom),player(jon) ==> winner(jon)),20),
             count((player(tom),player(jon) ==> ~winner(tom),~winner(jon)),30)])
...
| ?- show_sw
Switch exp1(jon): 1 (p: 0.60057034) 2 (p: 0.06536821) 3 (p: 0.33406143)
Switch exp1(tom): 1 (p: 0.08420895) 2 (p: 0.20973622) 3 (p: 0.70605482)
| ?- prob player(tom),player(jon) ==> winner(tom)
Probability of player(tom),player(jon)==>winner(tom) is: 0.499604
```
Introduction

CHRiSM

Implementation of CHRiSM

Discussion and conclusion

CHR(PRISM)

Jon Sneyers, Wannes Meert and Joost Vennekens

CHRiSM: Chance Rules induce Statistical Models
PRISM [Sato 1995, Sato & Kameya 1997]

- PRISM extends Prolog with probabilistic built-ins
- Implemented on top of B-Prolog [Zhou 1994-2009]
- Also several CHR systems available for B-Prolog
- Still, CHR(PRISM) does not work directly “out of the box”
  - reason: PRISM assumes pure Prolog
  - CHR implementations heavily use “dirty Prolog”
    (action rules, global variables, setarg, ...)
Current prototype uses toychr [Duck 2004]
  - naive implementation of CHR, very inefficient
  - uses only pure Prolog

Translation CHRIISM $\rightarrow$ CHR(PRISM) is more or less straightforward

Some details in the paper
Introduction

CHRiSM

Implementation of CHRiSM

Discussion and conclusion

Advantages of CHRiSM over PCHR

Related formalisms

Conclusion

Jon Sneyers, Wannes Meert and Joost Vennekens

CHRiSM: Chance Rules induce Statistical Models
Advantages of CHRiSM over PCHR

- CHRiSM semantics are more natural:
  - PCHR does not make sense for probabilistic propagation rules, CHRiSM does
  - chance rules have a local meaning, PCHR rules don’t

- CHRiSM semantics are more usable:
  - CHRiSM allows more execution control ($\omega_r$ and $\omega_p$ semantics)
  - CHRiSM should allow efficient implementation

- PRISM features can be used in CHRiSM
  - PCHR has only sampling, no learning etc.
  - no need to reinvent the wheel
CP-logic (LPADs) [Vennekens et al. 2006] can be encoded in CHRIISM. Details in the paper.

Many other probabilistic logic programming formalisms are sublogics of CP-logic:

- PRISM itself
- ProbLog [De Raedt et al. 2007]
- ICL [Poole 1997]

Bayesian network-inspired formalisms:

- BLP [Kersting & De Raedt 2007] covered by CHRIISM
- Others are more difficult (they support more complicated distributions):
  - RBN [Jaeger 1997]
  - CLP(BN) [Santos Costa et al. 2008]
  - Blog [Milch et al. 2007]
Introduction

CHRiSM

Implementation of CHRiSM

Discussion and conclusion

Advantages of CHRiSM over PCHR

Related formalisms

Conclusion

Summary

- New way to add probabilities to CHR: CHRiSM
  - based on CHR(PRISM)
  - has advantages over PCHR
- Naive prototype implementation:
- Subsumes other probabilistic-logic formalisms
- Exploratory work: still a lot to be done!
Future work

- Language design
  - current syntax/semantics good in practice?
  - need to investigate much more examples
- Efficient implementation
  - essential for feasible learning
  - PRISM uses tabling, cf. work on CHR+tabling (e.g. in XSB)
  - perhaps consider CHR(ProbLog) too?
- Currently: only ground goals/results $\rightarrow$ extend to non-ground
- Notion of probabilistic termination
  - program can terminate probabilistically but not classically
  - no problem for sampling
  - problem (of PRISM) for probability computation / learning
- Declarative semantics for CHriSM
  - based on distribution semantics of PRISM?
  - direct model semantics for CHriSM?
  - soundness/completeness of operational sem. w.r.t. decl. sem.
Questions?