

CHRiSM: CHANCE RULES INDUCE STATISTICAL MODELS

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

驰

CHR'09 Workshop
Pasadena, California, USA, July 2009



- 1 Introduction
 - Constraint Handling Rules
 - PCHR
- 2 CHRiSM
- 3 Implementation of CHRiSM
- 4 Discussion and conclusion

Constraint Handling Rules [Frühwirth 1991]

3/27

- ▶ 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

4/27

- ▶ Declarative semantics:
 - ▶ classical logic semantics
 - ▶ linear logic semantics [Betz & Frühwirth 2005]
- ▶ Abstract operational semantics ω_t
- ▶ Refined operational semantics ω_r [Duck et al. 2004]
 - ▶ activate constraints depth-first, left-to-right
 - ▶ search for matching rules by trying occurrences in textual order
- ▶ Priority semantics ω_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: ω_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

6/27

- ▶ 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 ω_t , but its semantics depends on the full non-determinism of ω_t
 - ▶ 'refined' or 'priority' PCHR is probably not desirable
 - ▶ efficient implementation of PCHR is not straightforward



1 Introduction

2 CHRiSM

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

3 Implementation of CHRiSM

4 Discussion and conclusion

- ▶ 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



Syntax of chance rules

9/27

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

Syntax: rule with probability Prob

Prob ?? Head \Leftrightarrow 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

Syntax of chance rules

9/27

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

Syntax: rule with probability Prob

Prob ?? Head \Leftrightarrow 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

Syntax of chance rules

9/27

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

Syntax: rule with probability Prob

Prob ?? Head \Leftrightarrow 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

Probability expressions

10/27

- ▶ Different kinds of probability expressions Prob allowed:
 - ▶ numbers:
 - head:0.5 ; tail:0.5
 - ▶ arithmetic expression which is ground at runtime:
 - eval(1-K) ?? maybe_keep_me(K) <=> true.
 - ▶ probabilities are unknown:
 - roll <=> ?? 1 ; 2 ; 3 ; 4 ; 5 ; 6
 - ▶ probabilities are unknown and parametrized:
 - roll(Die) <=> Die ?? 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

12/27

CHRiSM program

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

Example interaction

```
| ?- sample toss
```

Example 1: coin toss

12/27

CHRiSM program

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

Example interaction

```
| ?- sample toss
toss <=> tail.
| ?- sample toss,toss
toss <=> tail.
| ?- sample toss,toss
toss <=> tail.
```

Example 1: coin toss

12/27

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

12/27

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

12/27

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

12/27

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

12/27

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

12/27

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

13/27

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)
```

Example 2: rock-scissors-paper

13/27

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 2: rock-scissors-paper

13/27

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 2: rock-scissors-paper

13/27

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 2: rock-scissors-paper

13/27

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 2: rock-scissors-paper

13/27

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 2: rock-scissors-paper

13/27

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

14/27

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

Example 3: generating random graphs

14/27

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

14/27

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

14/27

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

14/27

CHRiSM program

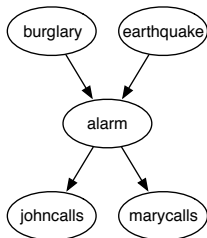
```
0.5 ?? node(A), node(B) ==> edge(A,B).
```

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 4: Bayesian networks

15/27



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).

```

Features of PRISM

16/27

- ▶ PRISM has many nice features, a.o.:
 - ▶ Probabilistic execution (`sample`)
 - ▶ Probability computation (`prob`)
 - ▶ EM-learning (`learn`)
- ▶ These features can also be used in CHRiSM

PRISM features in CHRiSM

17/27

- ▶ 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: `prob goal <==> result`
 - ▶ compute probability that “`sample goal`” gives “`result`”
 - ▶ `prob goal ===> result`
compute probability that “`sample goal`” gives something of the form “`result, otherstuff`”
- ▶ EM-learning: `learn(observations)`
 - ▶ `observations`: a list of observations of the form “`goal <==> result`” or “`goal ===> result`”
 - ▶ compute an assignment to the unknown probabilities such that the likelihood of the observations is maximized

Example: learning

18/27

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 expl(jon): 1 (pr: 0.60057034) 2 (pr: 0.06536821) 3 (pr: 0.33406143)
Switch expl(tom): 1 (pr: 0.08420895) 2 (pr: 0.20973622) 3 (pr: 0.70605482)
| ?- prob player(tom),player(jon) ==> winner(tom)
Probability of player(tom),player(jon) ==> winner(tom) is: 0.499604

```

Example: learning

18/27

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 exp(jon): 1 (p: 0.8567034) 2 (p: 0.06536821) 3 (p: 0.33406143)

```

Example: learning

18/27

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

18/27

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

18/27

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

18/27

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

18/27

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
```



- 1 Introduction
- 2 CHRiSM
- 3 Implementation of **CHRiSM**
 - PRISM
 - CHR(PRISM)
- 4 Discussion and conclusion

PRISM [Sato 1995, Sato & Kameya 1997]

20/27

- ▶ 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 CHRiSM \rightarrow CHR(PRISM) is more or less straightforward
- ▶ Some details in the paper



- 1 Introduction
- 2 CHRiSM
- 3 Implementation of CHRiSM
- 4 Discussion and conclusion
 - Advantages of **CHRiSM** over PCHR
 - Related formalisms
 - Conclusion

Advantages of CHRiSM over PCHR

23/27

- ▶ 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 (ω_r and ω_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

Related formalisms

24/27

- ▶ CP-logic (LPADs) [Vennekens et al. 2006] can be encoded in CHRiSM
 - ▶ 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 CHRiSM
 - ▶ Others are more difficult (they support more complicated distributions):
 - ▶ RBN [Jaeger 1997]
 - ▶ CLP(BN) [Santos Costa et al. 2008]
 - ▶ Blog [Milch et al. 2007]

- ▶ New way to add probabilities to CHR: CHRiSM
 - ▶ based on CHR(PRISM)
 - ▶ has advantages over PCHR
- ▶ Naive prototype implementation:
<http://www.cs.kuleuven.be/~jon/chris>
- ▶ Subsumes other probabilistic-logic formalisms
- ▶ Exploratory work: still a lot to be done!

- ▶ 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 → 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?*