

Turing-Complete Subclasses of CHR

Jon Sneyers

December 11th, 2008



DTAI



KATHOLIEKE UNIVERSITEIT
LEUVEN

驰

CHR TEAM

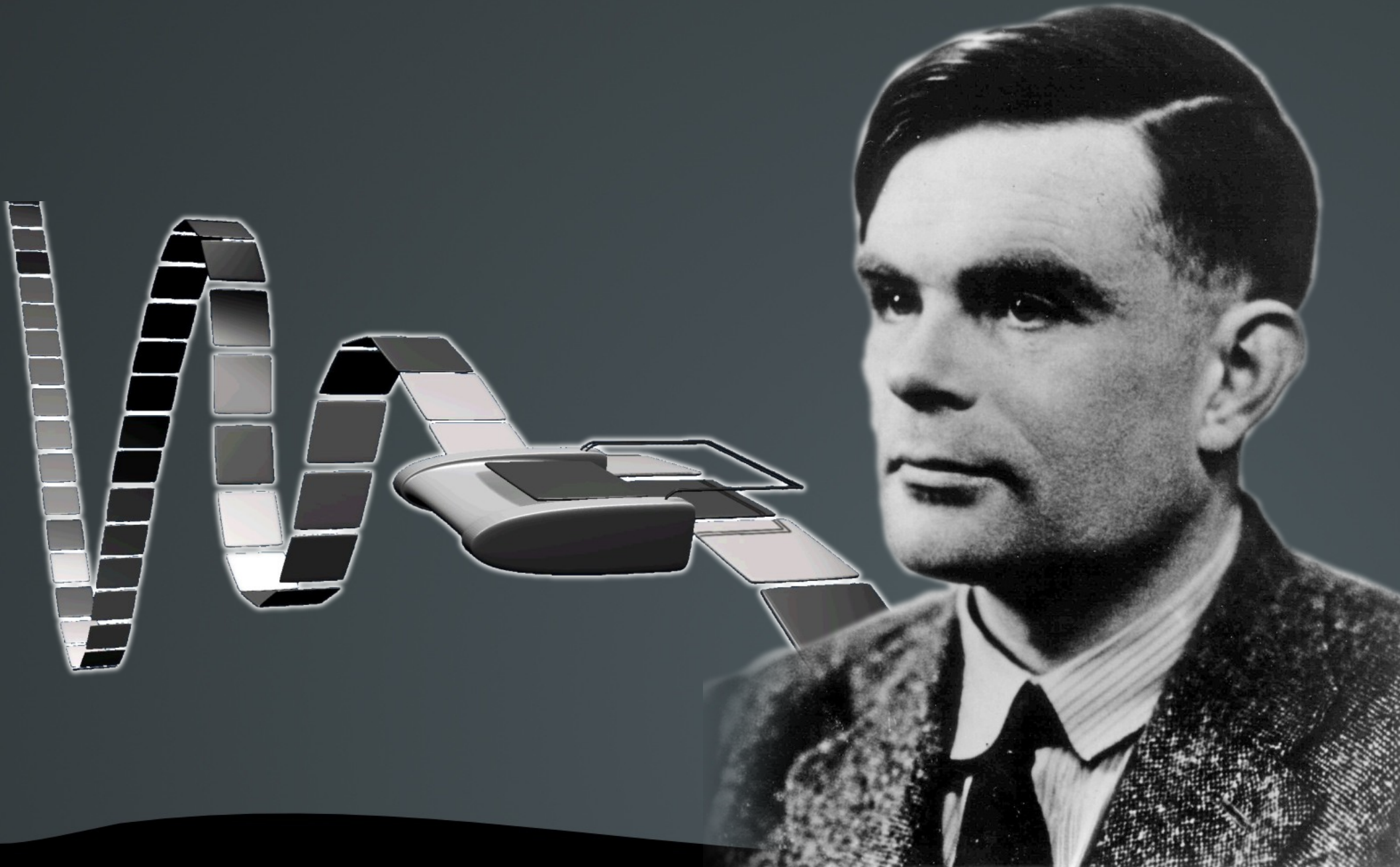
Constraint Handling Rules

驰

cf. tutorial by Tom Schrijvers on first day of ICLP'08



Turing-completeness



CHR is Turing-complete...

... but which (syntactic) **subclasses** of CHR are still Turing-complete and which are not?

in other words: which features of CHR are really needed and which are redundant?
(from the computability point of view)



Host language

- CHR extends host language (e.g. Prolog); constraint arguments are of host language data type
- **Propositional CHR**: no host language, no arguments (only arity 0 constraints)
- **CHR-only**: no host language, arguments are constants or variables (that cannot be bound)
- **Sufficiently strong** host language: complex terms are available, we can do **arithmetic**



propositional CHR
(no constraint arguments)



CHR-only
(no host language)

CHR with
sufficiently strong
host language

CHR with
Turing-complete
host language



not Turing-complete

propositional CHR
(no constraint arguments)



CHR-only
(no host language)

CHR with
sufficiently strong
host language

CHR with
Turing-complete
host language



Turing-complete





propositional CHR
(no constraint arguments)



CHR-only
(no host language)

CHR with
sufficiently strong
host language



CHR with
Turing-complete
host language



CHR'05 result:
RAM machine
simulator in CHR



TMSIM:

```
r1 @ delta(Q,S,P,T,left), adj(L,C) \ state(Q), cell(C,S), head(C)
    <=> state(P), cell(C,T), head(L).
r2 @ delta(Q,S,P,T,right), adj(C,R) \ state(Q), cell(C,S), head(C)
    <=> state(P), cell(C,T), head(R).
r3 @ delta(Q,S,P,T,left) \ left(C), state(Q), cell(C,S), head(C)
    <=> cell(L,b), left(L), adj(L,C), state(P), cell(C,T), head(L).
r4 @ delta(Q,S,P,T,right) \ right(C), state(Q), cell(C,S), head(C)
    <=> cell(R,b), adj(C,R), right(R), state(P), cell(C,T), head(R).
```

also in CHR'05 paper:
Turing machine
simulator in CHR-only

propositional CHR
(no constraint arguments)



CHR-only
(no host language)



CHR with
sufficiently strong
host language









CHR with
Turing-complete
host language











Kinds of rules

- Different kinds of rules:
 - **propagation** rules: $KeptHead \Rightarrow Body$
 - **simplification** rules: $RemovedHead \Leftrightarrow Body$
 - simpagation rules: $KeptHead \setminus RemovedHead \Leftrightarrow Body$
- Simplification rules can simulate all rules
“only simplification rules” = “only simpagation rules” = “any kind of rule”
- “Programs with only propagation rules” vs
“Programs with any kind of rule”



	prop. -only	simp. -only
propositional CHR (no constraint arguments)		
CHR-only (no host language)		
CHR with <i>sufficiently strong</i> host language		
CHR with <i>Turing-complete</i> host language		



	prop. -only	simp. -only
propositional CHR (no constraint arguments)		
CHR-only (no host language)		
CHR with <i>sufficiently strong</i> host language		
CHR with <i>Turing-complete</i> host language		



TMSIM-PROP:

```
% add timestamps
head(C) ==> inittime(T), head(T,C).
inittime(T), state(Q) ==> state(T,Q).
inittime(T), cell(C,S) ==> cell(T,C,S).
inittime(T), adj(L,R) ==> adj(T,L,R).

% compute next step
r13 @ state(T,Q), head(T,C), cell(T,C,S), delta(Q,S,Q2,S2,left)
    ==> next(T,U), state(U,Q2), cell(U,C,S2), mleft(T,C,U), cright(T).
r24 @ state(T,Q), head(T,C), cell(T,C,S), delta(Q,S,Q2,S2,right)
    ==> next(T,U), state(U,Q2), cell(U,C,S2), mright(T,C,U), cleft(T).
state(T,Q), head(T,C), cell(T,C,S), nodelta(Q,S), reject(Q) ==> fail.

% move head, extending tape if needed
mleft(T,C,U), adj(T,L,C) ==> L \== null | head(U,L), cleft(T).
mleft(T,C,U), adj(T,null,C) ==> head(U,L), adj(U,null,L), adj(U,L,C).
mright(T,C,U), adj(T,C,R) ==> R \== null | head(U,R), cright(T).
mright(T,C,U), adj(T,C,null) ==> head(U,R), adj(U,C,R), adj(U,R,null).

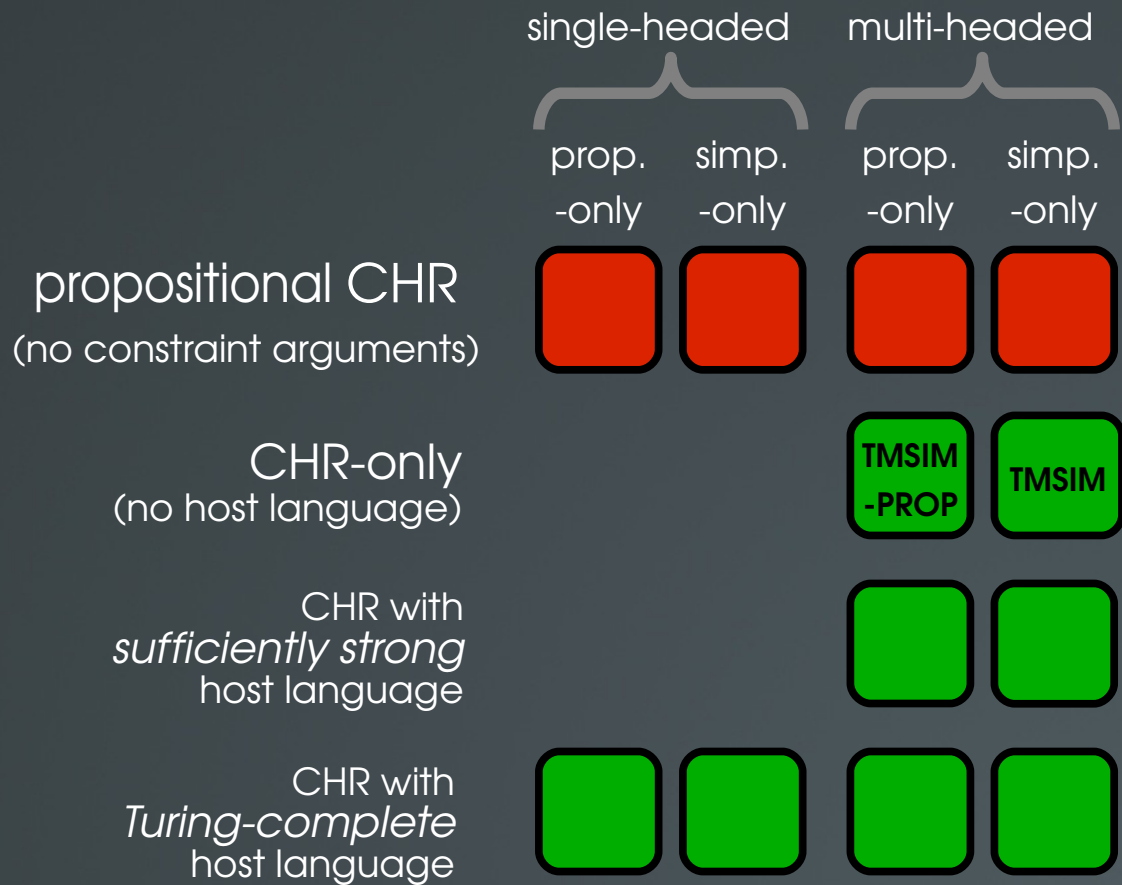
% copy non-modified tape to next timestamp
cell(T,C,S), next(T,U), head(T,C2) ==> C \== C2 | cell(U,C,S).
adj(T,L,R), next(T,U) ==> L \== null, R \== null | adj(U,L,R).
cleft(T), next(T,U), adj(T,X,null) ==> adj(U,X,null).
cright(T), next(T,U), adj(T,null,X) ==> adj(U,null,X).
```



Multi-headed vs. single-headed

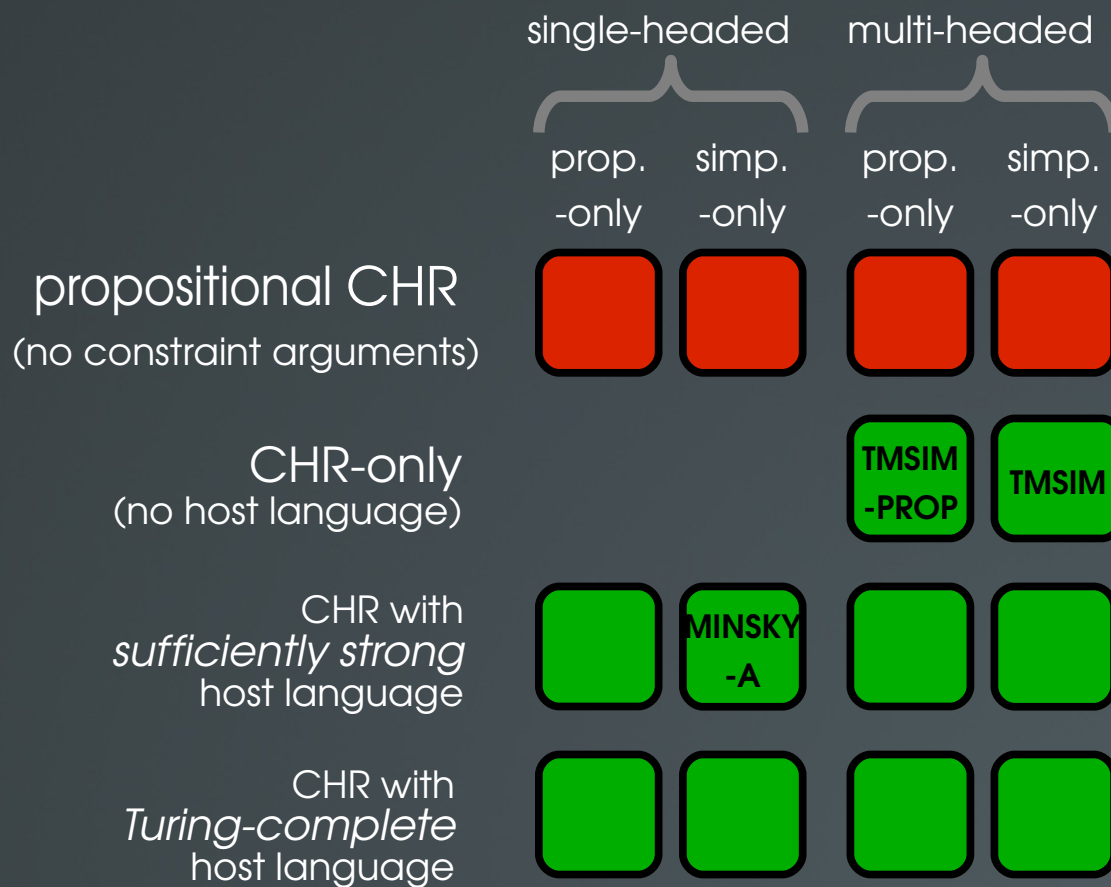
- Head of a rule is a *conjunction* of constraints
 - 1 conjunct: **single-headed** rule
 - 2 conjuncts: two-headed rule
 - > 1 conjuncts: **multi-headed** rule
- >2 -headed rule can be written using 2-headed rules:
 - “ $a, b, c, d \Rightarrow e$ ” can be written as follows:
“ $a, b \Rightarrow ab$ ” “ $ab, c \Rightarrow abc$ ” “ $abc, d \Rightarrow e$ ”





MINSKY-A:

see Di Giusto, Gabbrielli, Meo (2008): "Expressiveness of multiple heads in CHR"



MINSKY-A:

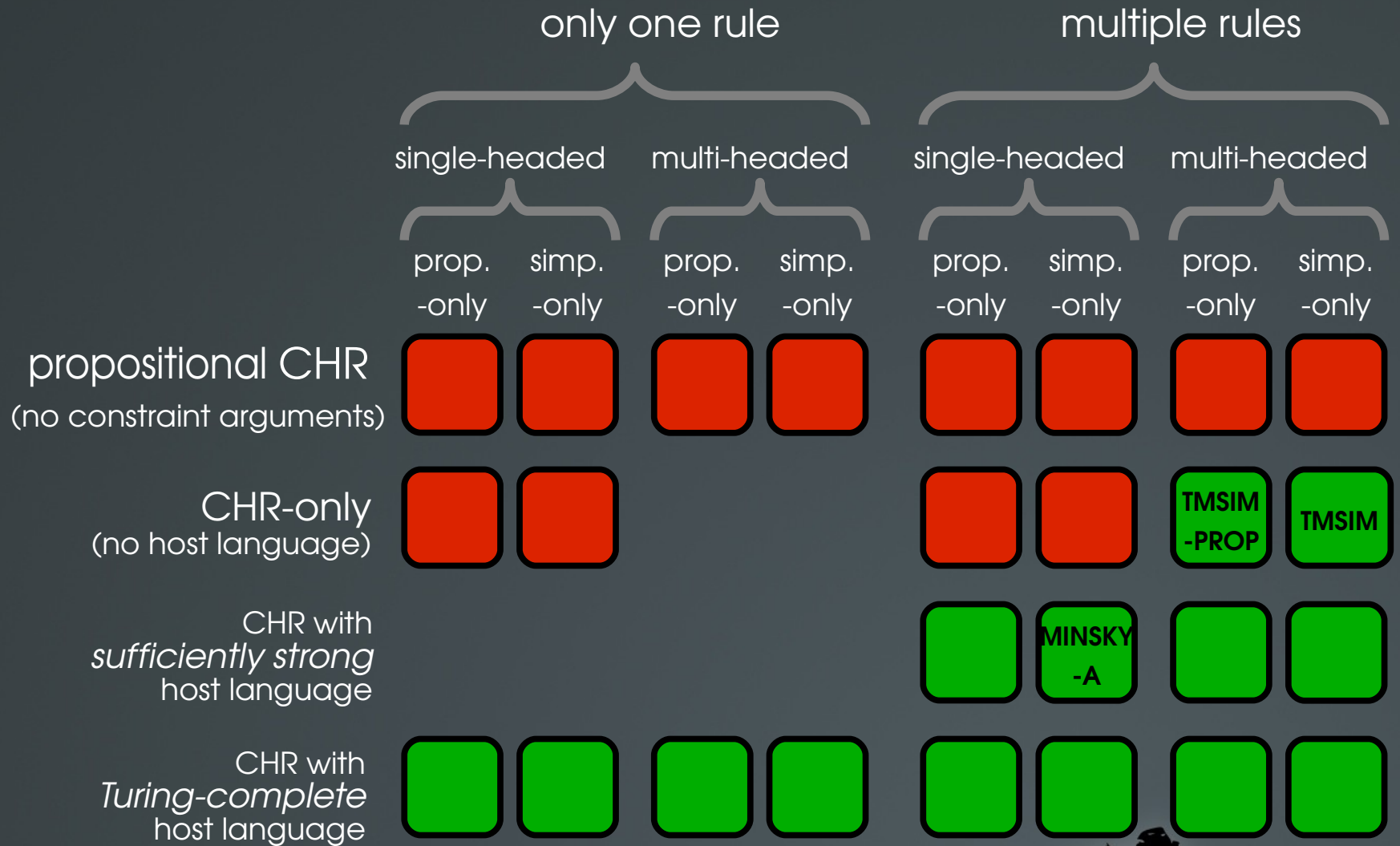
see Di Giusto, Gabrielli, Meo (2008): "Expressiveness of multiple heads in CHR"

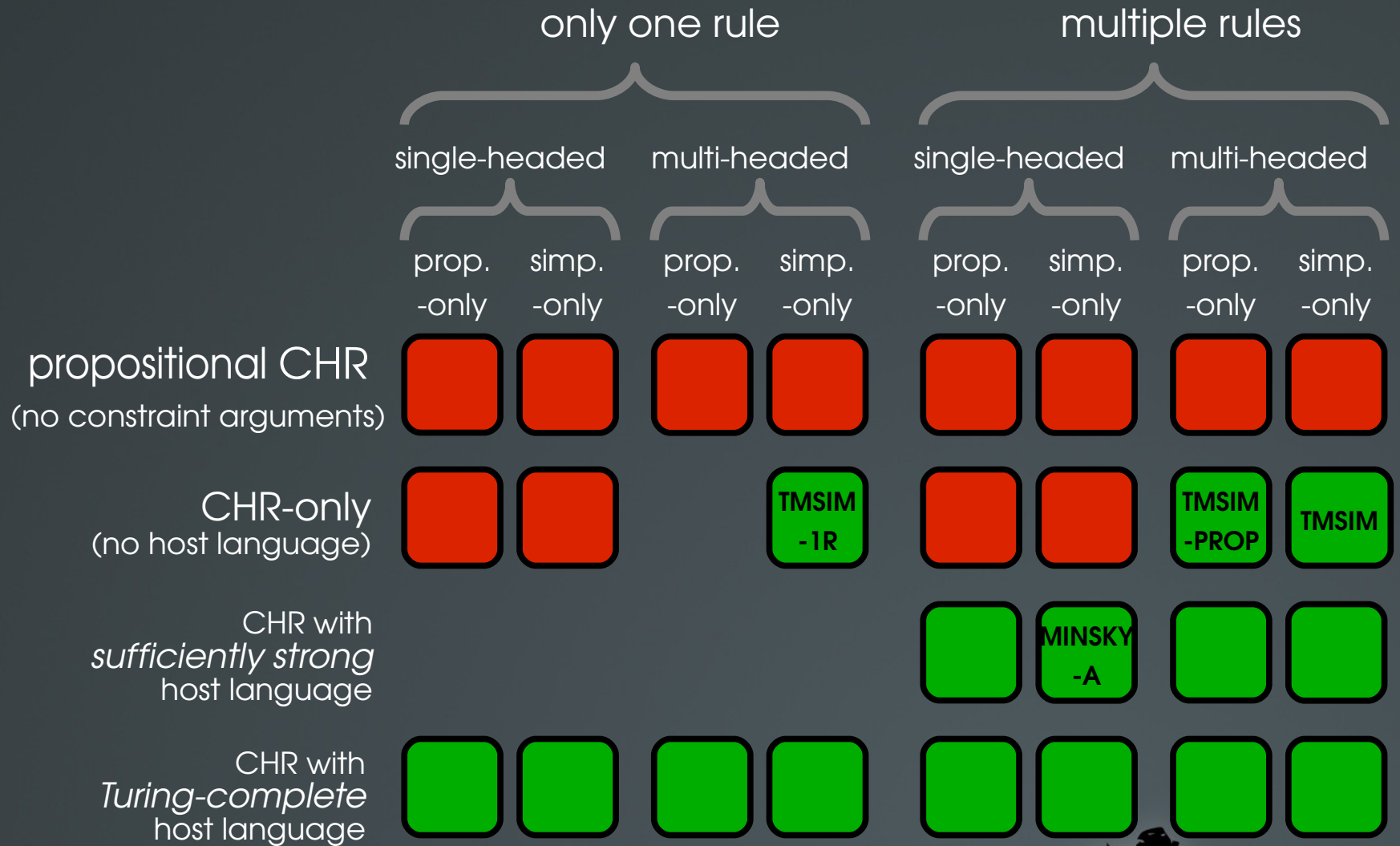


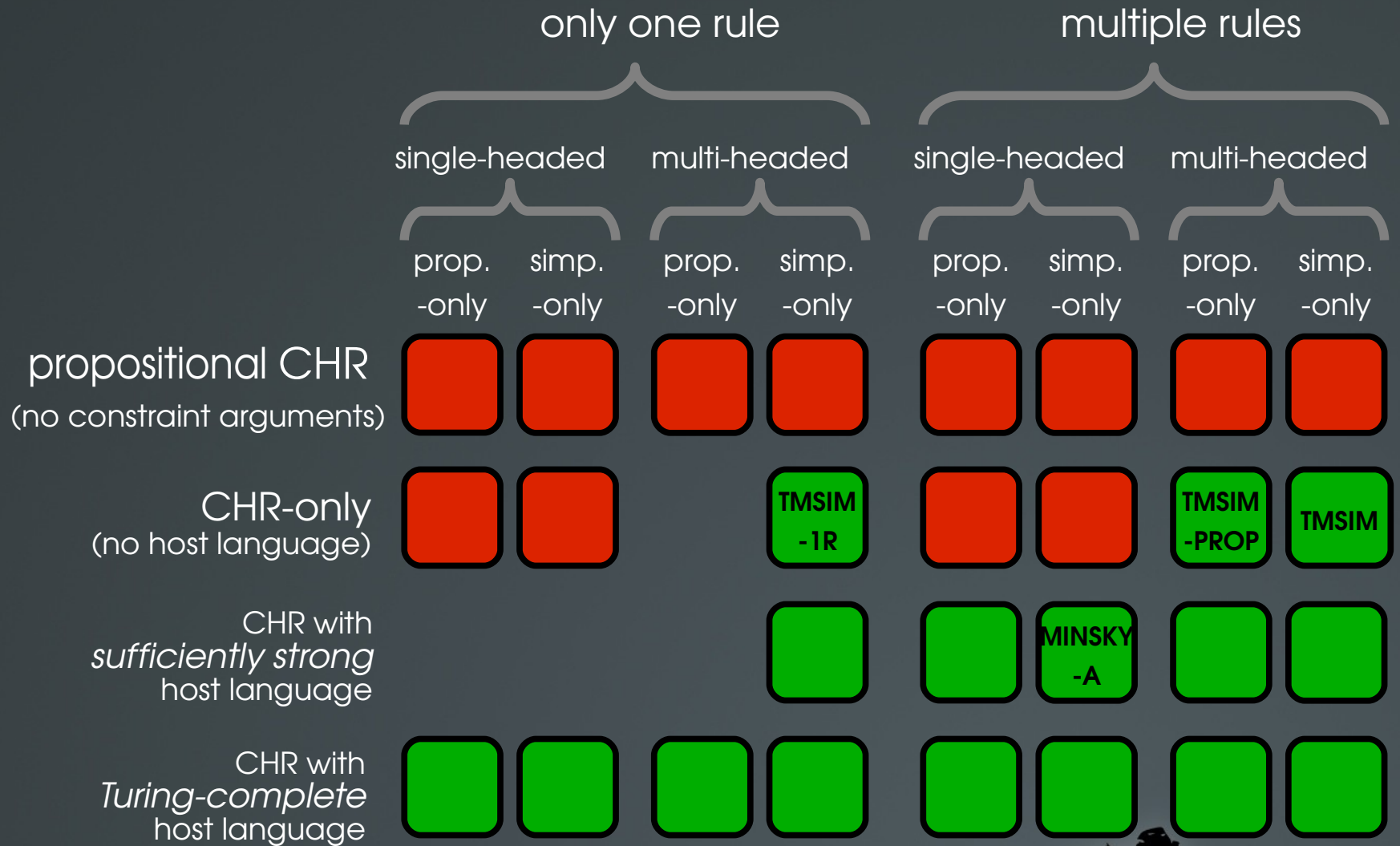
Number of rules

- Do we really need more than one rule?









TMSIM-2R:

```
% left(C) <=> adj(L,L), adj(L,C), cell(L,b).
% right(C) <=> adj(R,R), adj(C,R), cell(R,b).

r13 @ delta(Q,S,P,T,left), state(Q), head(C) \ adj(L,C), adj(C,R), cell(C,S)
      <=> adj(L,C2), adj(C2,R), adj(C,C), cell(C,b), cell(C2,T), state(P), head(L).
r24 @ delta(Q,S,P,T,right), state(Q), head(C) \ adj(L,C), adj(C,R), cell(C,S)
      <=> adj(L,C2), adj(C2,R), adj(C,C), cell(C,b), cell(C2,T), state(P), head(R).
```

TMSIM-1R:

```
% adj(A,B) <=> adj(A,B,left), adj(B,A,right).

r1234 @ delta(Q,S,P,T,D), state(Q), head(C) \ adj(A,C,D), adj(C,B,D),
                                               cell(C,S), adj(C,A,E), adj(B,C,E)
      <=> adj(A,C2,D), adj(C2,B,D), adj(C,C,D), cell(C,b), cell(C2,T),
          adj(C2,A,E), adj(B,C2,E), adj(C,C,E), state(P), head(A).
```



Operational semantics

- **abstract semantics**: rules are applied in any order (so program has to be confluent)
- **refined semantics**: specific execution order (cf. Prolog: left-to-right, depth-first + triggering / “active constraint”)



propositional
CHR

(no constraint arguments)

CHR-only
(no host language)

CHR with
sufficiently strong
host language

CHR with
Turing-complete
host language

only one rule

multiple rules

single-headed

multi-headed

single-headed

multi-headed

prop.
-only

simp.
-only

prop.
-only

simp.
-only

prop.
-only

simp.
-only

prop.
-only

simp.
-only

abstract semantics

refined semantics

TMSIM
-1R

TMSIM
-PROP

TMSIM

MINSKY
-A



only one rule

multiple rules

single-headed

multi-headed

single-headed

multi-headed

prop.
-only

simp.
-only

prop.
-only

simp.
-only

prop.
-only

simp.
-only

prop.
-only

simp.
-only

propositional
CHR

(no constraint
arguments)

abstract
semantics

refined
semantics

CHR-only
(no host language)

CHR with
sufficiently strong
host language

CHR with
Turing-complete
host language



only one rule

multiple rules

single-headed

multi-headed

single-headed

multi-headed

prop.
-only

simp.
-only

prop.
-only

simp.
-only

prop.
-only

simp.
-only

prop.
-only

simp.
-only

propositional
CHR

(no constraint
arguments)

abstract
semantics

refined
semantics

CHR-only
(no host language)

CHR with
sufficiently strong
host language

CHR with
Turing-complete
host language



Conclusion

- one multi-headed rule is enough
- single-headed rules are not enough (unless we have a sufficiently strong host language)
- propositional CHR is not enough, **unless** we can use the refined semantics (!)
- more information:
my PhD thesis (chapter 10.3)

