Answer Set Solving in Practice

Torsten Schaub[1]
University of Potsdam
torsten@cs.uni-potsdam.de

Potassco

[1] Standing on the shoulders of a great research group and community!

Systems: Overview

Outline

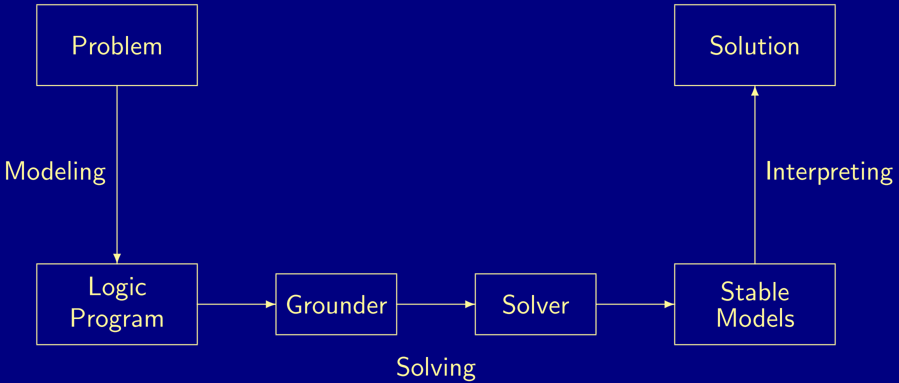
Potassco

- Potsdam Answer Set Solving Collection
- Home https://potassco.org[5]
- Training https://teaching.potassco.org
- Sources https://github.com/potassco
- License MIT License
- Mailing lists
  - https://sourceforge.net/projects/potassco/lists/potassco-users
  - https://sourceforge.net/projects/potassco/lists/potassco-announce
  - https://lists.cs.uni-potsdam.de/subscribe/krnews

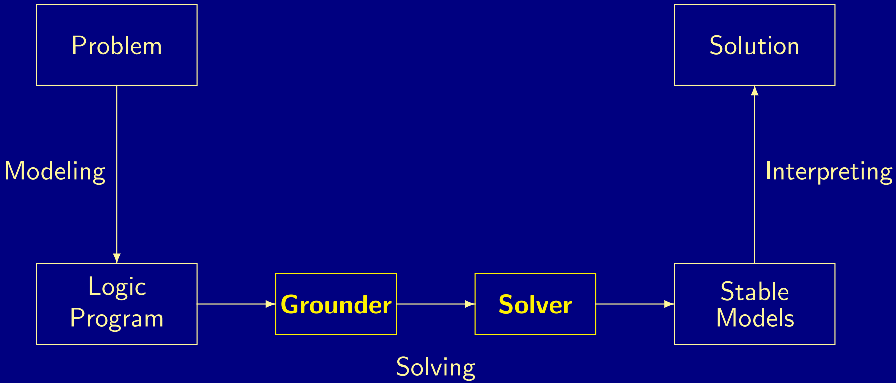
[5] Deprecated former site https://potassco.sourceforge.net

Outline

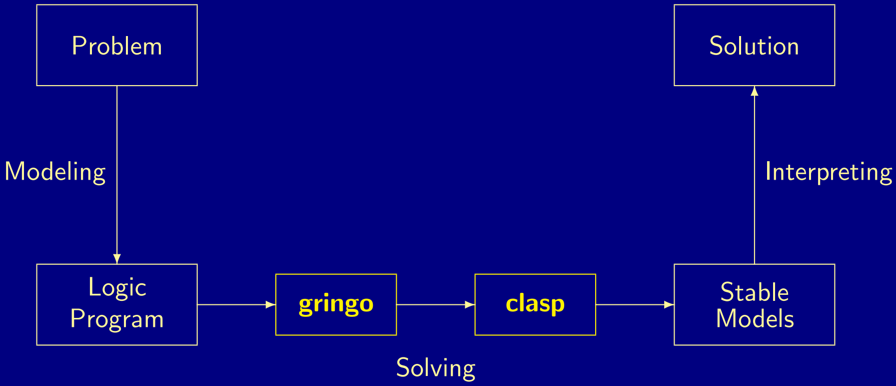
ASP solving process



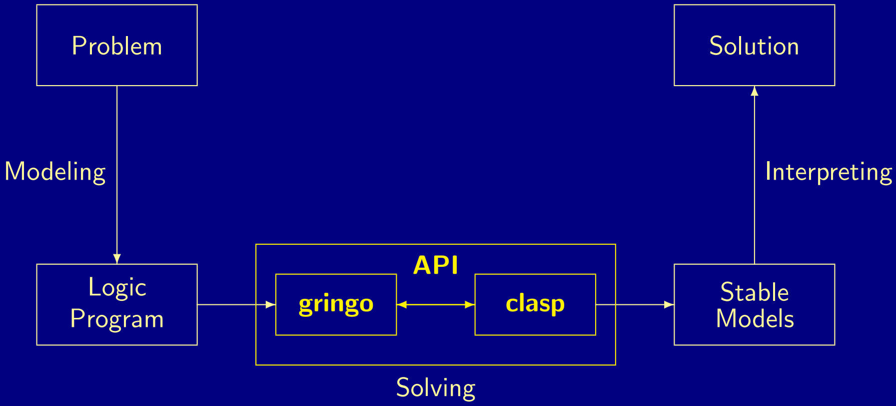
ASP solving process



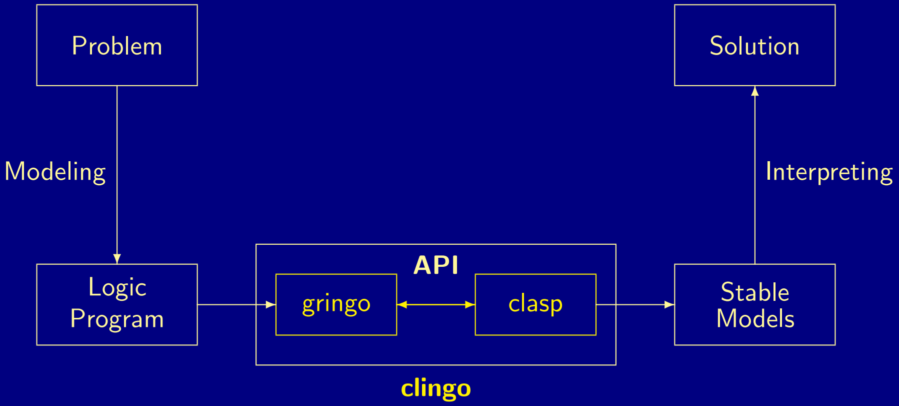
ASP solving process



ASP solving process



ASP solving process



gringo

- Idea systematically replace object variables by variable-free terms, applying stable-model preserving simplifications
- Features
  - expressive modeling language
  - procedural attachments
  - meta programming
- Technology semi-naive database evaluation
- References [84]

Language constructs

- Facts `q(42).`
- Rules `p(X) :- q(X), not r(X).`
- Conditional literals `p :- q(X) : r(X).`
- Disjunction `p(X) ; q(X) :- r(X).`
- Integrity constraints `:- q(X), p(X).`
- Choice `2 { p(X,Y) : q(X) } 7 :- r(Y).`
- Aggregates `s(Y) :- r(Y), 2 #sum{ X : p(X,Y), q(X) } 7.`

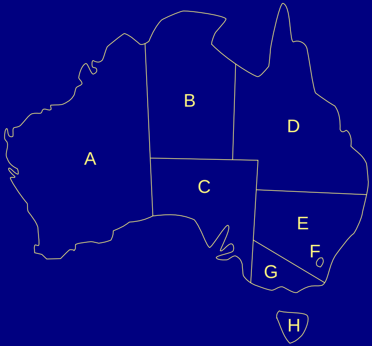
- Multi-objective optimization `:~ q(X), p(X,C). [C]`

  `#minimize { C : q(X), p(X,C) }`

Potassco

Example

Graph coloring




```
% instance
vertex(a;b;c;d;e;f;g;h).
edge(a,(b;c)).
edge(b,(c;d)).
edge(c,(d;e;g)).
edge(d,e).
edge(e,(f;g)).
color(red;green;blue).

% encoding
{ assign(V,C) : color(C) } = 1 :- vertex(V).
:- edge(U,V), assign(U,C), assign(V,C).
```

Grounding

```
$ gringo --text enconding.lp instance.lp
vertex(a). vertex(b). vertex(c). vertex(d). vertex(e). vertex(f). vertex(g). vertex(h).
edge(a,b). edge(a,c). edge(b,c). edge(b,d). edge(c,d).
edge(c,e). edge(c,g). edge(d,e). edge(e,f). edge(e,g).
color(red). color(green). color(blue).
{ assign(a,red); assign(a,green); assign(a,blue)} = 1.
{ assign(b,red); assign(b,green); assign(b,blue)} = 1.
{ assign(c,red); assign(c,green); assign(c,blue)} = 1.
{ assign(d,red); assign(d,green); assign(d,blue)} = 1.
{ assign(e,red); assign(e,green); assign(e,blue)} = 1.
{ assign(f,red); assign(f,green); assign(f,blue)} = 1.
{ assign(g,red); assign(g,green); assign(g,blue)} = 1.
{ assign(h,red); assign(h,green); assign(h,blue)} = 1.
:- assign(b,red),assign(a,red).     :- assign(b,green),assign(a,green).
:- assign(b,blue),assign(a,blue).   :- assign(c,red),assign(a,red).
:- assign(c,green),assign(a,green). :- assign(c,blue),assign(a,blue).
:- assign(c,red),assign(b,red).     :- assign(c,green),assign(b,green).
:- assign(c,blue),assign(b,blue).   :- assign(d,red),assign(b,red).
:- assign(d,green),assign(b,green). :- assign(d,blue),assign(b,blue).
:- assign(d,red),assign(c,red).     :- assign(d,green),assign(c,green).
:- assign(d,blue),assign(c,blue).   :- assign(e,red),assign(c,red).
:- assign(e,green),assign(c,green). :- assign(e,blue),assign(c,blue).
:- assign(g,red),assign(c,red).     :- assign(g,green),assign(c,green).
:- assign(g,blue),assign(c,blue).   :- assign(e,red),assign(d,red).
:- assign(e,green),assign(d,green). :- assign(e,blue),assign(d,blue).
:- assign(f,red),assign(e,red).     :- assign(f,green),assign(e,green).
:- assign(f,blue),assign(e,blue).   :- assign(g,red),assign(e,red).
:- assign(g,green),assign(e,green). :- assign(g,blue),assign(e,blue).
```

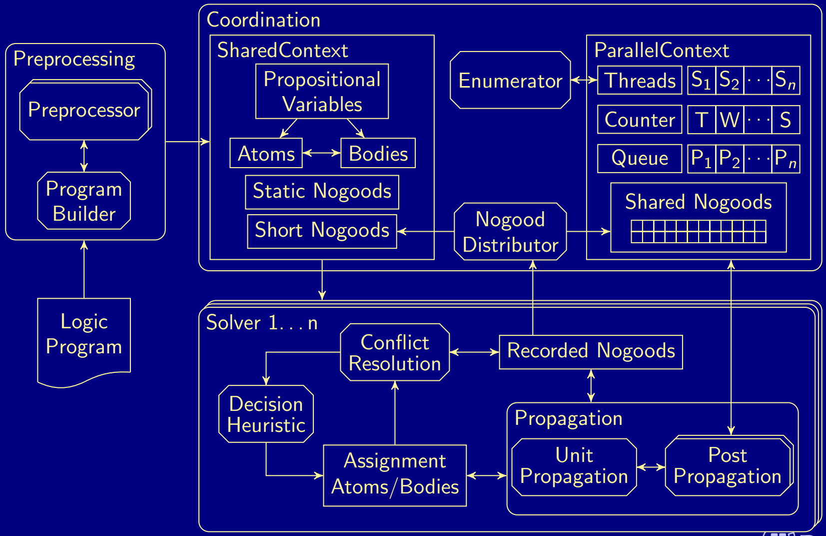
Grounding for the solver

```
$ gringo enconding.lp instance.lp
asp 1 0 0
1 0 1 1 0 0
1 0 1 2 0 0
...
1 0 0 0 2 30 31
1 0 0 0 2 32 33
...
1 1 3 31 33 35 0 1 22
1 0 1 51 1 1 3 31 1 33 1 35 1
1 0 1 52 1 2 3 31 1 33 1 35 1
1 0 1 53 0 2 51 -52
1 0 0 0 2 22 -53
...
4 10 color(red) 0
4 12 color(green) 0
...
4 13 assign(a,red) 1 31
4 13 assign(b,red) 1 30
...
4 9 vertex(a) 0
4 9 vertex(b) 0
...
4 9 edge(a,b) 0
4 9 edge(a,c) 0
...
0
```

clasp

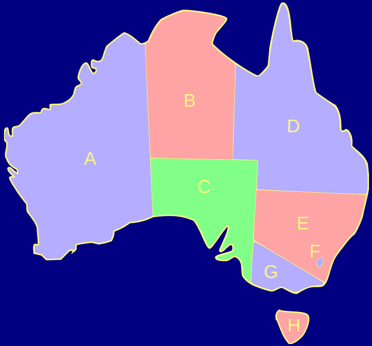
- Idea a versatile, high-performant solver for ASP, OPB, SAT
- Features
  - multi-threaded architecture
  - multi-objective optimization
  - various reasoning modes
- Technology conflict-driven constraint learning
- References [62, 63, 67]
- Applications [98] and many more within *clingo*

Multi-threaded architecture of *clasp*



Example

Graph coloring



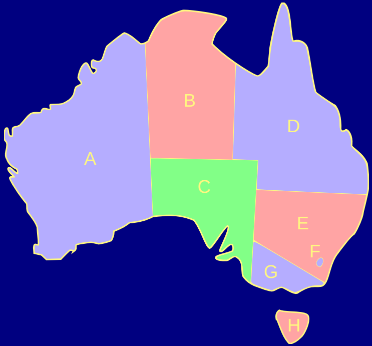

```
$ clasp grounding.aspif
clasp version 3.3.8
Reading from grounding.aspif
Solving...
Answer: 1
vertex(a) vertex(b) vertex(c) ...
edge(a,b) edge(a,c) edge(b,c) ...
color(red) color(green) color(blue)
assign(a,blue)  assign(b,red)
assign(c,green) assign(d,blue)
assign(e,red)   assign(g,blue)
assign(f,blue)  assign(h,red)
SATISFIABLE

Models    : 1+
Calls     : 1
Time      : 0.001s (Solving: 0.00s 1st Model: 0.00s Unsat: 0.00s)
CPU Time  : 0.001s
```

clingo

- Idea *clingo* = *gringo* + *clasp* + API
- APIs C, C++, Java, Lua, Prolog, Python, Rust, . . .
- Features
  - encompassing ASP system
  - multi-shot solving
  - theory solving
- References [69, 83, 86]
- Applications
  [95, 60, 52, 54, 17, 109, 42, 99, 111, 40, 47, 117, 16, 3, 15, 82] etc
- Systems [26, 8, 108, 114, 49, 11, 104, 51] etc

Example

Graph coloring




```
$ clingo encoding.lp instance.lp
clingo version 5.6.4
Reading from encoding.lp ...
Solving...
Answer: 1
vertex(a) vertex(b) vertex(c) ...
edge(a,b) edge(a,c) edge(b,c) ...
color(red) color(green) color(blue)
assign(a,blue)  assign(b,red)
assign(c,green) assign(d,blue)
assign(e,red)   assign(g,blue)
assign(f,blue)  assign(h,red)
SATISFIABLE

Models     : 1+
Calls      : 1
Time       : 0.001s (Solving: 0.00s 1st Model: 0.00s Unsat: 0.00s)
CPU Time   : 0.001s
```

Outline

Potassco

metasp

- Idea extend ASP via meta programming
- Features
  - reification
  - meta programming
  - ASP solver in a dozen lines
- References [61, 86]
- Applications [24, 48]

Grounding for meta programming

```
$ gringo --output=reify enconding.lp instance.lp
atom_tuple(0).
atom_tuple(0,1).
literal_tuple(0).
rule(disjunction(0),normal(0)).
atom_tuple(1).
atom_tuple(1,2).
rule(disjunction(1),normal(0)).
atom_tuple(2).
atom_tuple(2,3).
rule(disjunction(2),normal(0)).
atom_tuple(3).
atom_tuple(3,4).
rule(disjunction(3),normal(0)).
atom_tuple(4).
atom_tuple(4,5).
rule(disjunction(4),normal(0)).
atom_tuple(5).
atom_tuple(5,6).
rule(disjunction(5),normal(0)).
atom_tuple(6).
atom_tuple(6,7).
rule(disjunction(6),normal(0)).
atom_tuple(7).
atom_tuple(7,8).
rule(disjunction(7),normal(0)).
atom_tuple(8).
atom_tuple(8,9).
...
```

Meta encoding, or ASP in ASP

```
conjunction(B) :- literal_tuple(B),
        hold(L) : literal_tuple(B, L), L > 0;
    not hold(L) : literal_tuple(B,-L), L > 0.

body(normal(B)) :- rule(_,normal(B)), conjunction(B).
body(sum(B,G))  :- rule(_,sum(B,G)),
    #sum { W,L :     hold(L), weighted_literal_tuple(B, L,W), L > 0 ;
           W,L : not hold(L), weighted_literal_tuple(B,-L,W), L > 0 } >= G.

  hold(A) : atom_tuple(H,A)   :- rule(disjunction(H),B), body(B).
{ hold(A) : atom_tuple(H,A) } :- rule(     choice(H),B), body(B).

#show.
#show T : output(T,B), conjunction(B).
```

clingcon

- Idea extend *clingo* with linear constraints over integers
- Features
  - integer variables not subject to grounding
  - large integer domains
  - basic constraints: `&sum` and `&distinct`
  - multi-objective optimization `&minimize`
- References [102, 10]
- Applications incorporate quantities, like resources and/or time

send+more=money

	s	e	n	d
+	m	o	r	e
m	o	n	e	y

Each letter corresponds exactly to one digit and all variables have to be pairwise distinct

	9	5	6	7
+	1	0	8	5
1	0	6	5	2

The example has exactly one solution

$$\{\ s \mapsto 9, e \mapsto 5, n \mapsto 6, d \mapsto 7, m \mapsto 1, o \mapsto 0, r \mapsto 8, y \mapsto 2\ \}$$

send+more=money

	s	e	n	d
+	m	o	r	e
m	o	n	e	y

Each letter corresponds exactly to one digit and all variables have to be pairwisely distinct

	9	5	6	7
+	1	0	8	5
1	0	6	5	2

The example has exactly one solution

$$\{\ s \mapsto 9, e \mapsto 5, n \mapsto 6, d \mapsto 7, m \mapsto 1, o \mapsto 0, r \mapsto 8, y \mapsto 2\ \}$$

send+more=money

```
                        sum(4,m).
arg(1,3,s). arg(2,3,m). sum(3,o).
arg(1,2,e). arg(2,2,o). sum(2,n).
arg(1,1,n). arg(2,1,r). sum(1,e).
arg(1,0,d). arg(2,0,e). sum(0,y).

digit(D) :- arg(_,_,D).
digit(D) :-   sum(_,D).

&dom { 0..9 } = D :- digit(D).

&sum {  M*D : arg(I,N,D), M=10**N;
       -M*D :   sum(N,D), M=10**N  } = 0.

&sum { E } > 0 :- (_,E) = #max { (N,D): sum(N,D) }.

&distinct { D : digit(D) }.

&show { D : digit(D) }.
```

fclingo

- Idea extend *clingo* with ASP-like linear constraints over integers
- Features
  - integer variables not subject to grounding
  - values of variables must be derivable, variables may be undefined
  - optional variables, defaults, assignments
  - check whether variables are defined
  - basic constraints: strict and non-strict `&sum`, `&min`, `&max`
- Technology translation to *clingcon*
- References [30, 29]
- Applications incorporate quantities, like resources and/or time

Potassco

Example

Configuration

```
price(frame,15).
price(bag,5).

  select(frame).
{ select( bag ) }.

&sum{V} = price(P) :- select(P), price(P,V).
&sum{price(P) : select(P)} = price(total).

#show select/1.
&show{price/1}.
```

Example

Configuration

```
$ fclingo 0 configuration-optionality.lp
fclingo version 0.1
Reading from ...configuration-optionality.lp
Solving...
Answer: 1
select(frame) \
val(price(total),15) val(price(frame),15)
Answer: 2
select(frame) select(bag) \
val(price(total),20) val(price(bag),5) val(price(frame),15)
SATISFIABLE

Models       : 2
Calls        : 1
Time         : 0.008s (Solving: 0.00s 1st Model: 0.00s Unsat: 0.00s)
CPU Time     : 0.008s
```

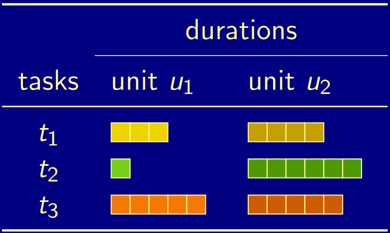
clingo[DL]

- Idea extend *clingo* with difference constraints over integers
- Features
  - integer variables not subject to grounding
  - a difference constraint '$x - y \leq c$'
    is represented as '`&diff(x-y) <= c`'
  - yields canonical witnessing integer assignments
- References [83, 86]
- Applications [1, 81, 103]

Potassco

Example

Task scheduling



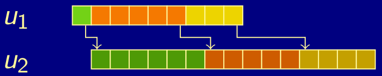

```
% instance

bound(16).  unit(1).     unit(2).
task(1).    dur(1,1,3).  dur(1,2,4).
task(2).    dur(2,1,1).  dur(2,2,6).
task(3).    dur(3,1,5).  dur(3,2,5).

% encoding

{assign(S,T): task(S)}=1 :- task(T).
{assign(S,T): task(T)}=1 :- task(S).

&diff { (S,U)-(S+1,U) } <= -D  :- assign(S,T), dur(T,U,D), task(S+1).
&diff { (S,U)-(S,U+1) } <= -D  :- assign(S,T), dur(T,U,D), unit(U+1).
&diff {     0-(S,U)   } <= 0   :- assign(S,T), unit(U).
&diff { (S,U)-0       } <= B-D :- assign(S,T), dur(T,U,D), bound(B).
```

Example

Task scheduling

```
$ clingo-dl version 1.4.0
Reading from example.lp
Solving...
Answer: 1
bound(16) unit(1) unit(2)
task(1) task(2) task(3)
dur(1,1,3) dur(1,2,4) dur(2,1,1)
dur(2,2,6) dur(3,1,5) dur(3,2,5)
assign(1,2) dl((1,1),0) dl((1,2),1)
assign(2,3) dl((2,1),1) dl((2,2),7)
assign(3,1) dl((3,1),6) dl((3,2),12)
SATISFIABLE

Models       : 1+
Calls        : 1
Time         : 0.039s (Solving: 0.00s 1st Model: 0.00s Unsat: 0.00s)
CPU Time     : 0.002s
```

unit	solution
	$t_2 < t_3 < t_1$



clingo[LP]

- Idea extend *clingo* with linear constraints over integers and/or reals
- Features
  - integer/real variables not subject to grounding
  - basic constraint `&sum`
  - optimization `&minimize`
  - integrates MILP solver (*lpsolve*, *cplex*, ...)
- References [83, 86]
- Applications notably, hybrid metabolic network completion [59]

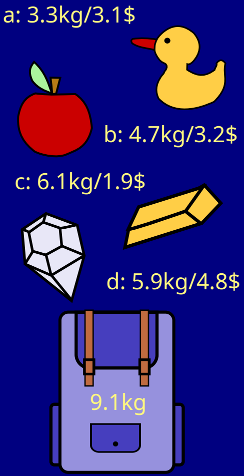
clingo[LPX]

- Idea extend *clingo* with linear constraints over rationals
- Features
  - rational variables not subject to grounding
  - basic constraints `&sum`
  - optimization `&minimize`
  - customized propagator based on simplex method
  - arbitrary precision arithmetics
- Applications notably, hybrid metabolic network completion [59]

Example

0/1 Knapsack

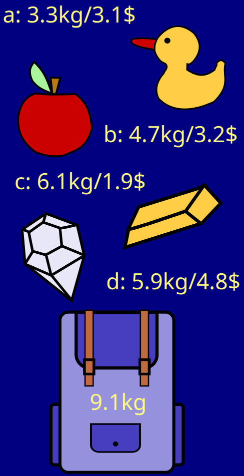
```
item(a;b;c;d).
weight(a,"3.3";b,"4.7";c,"6.1";d,"5.9").
value(a,"3.1";b,"3.2";c,"1.9";d,"4.8").
load("9.1").


{ pack(I) } :- item(I).

&sum { I } = 1 :- pack(I).
&sum { I } = 0 :- item(I), not pack(I).

&sum { W*I: weight(I,W) } <= L :- load(L).
&maximize { P*I: value(I,P) }.
```



Example

0/1 Knapsack

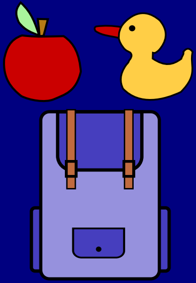
```
item(a;b;c;d).
weight(a,"3.3";b,"4.7";c,"6.1";d,"5.9").
value(a,"3.1";b,"3.2";c,"1.9";d,"4.8").
load("9.1").


{ pack(I) } :- item(I).

&sum { I } = 1 :- pack(I).
&sum { I } = 0 :- item(I), not pack(I).

&sum { W*I: weight(I,W) } <= L :- load(L).
&maximize { P*I: value(I,P) }.
```



Example

0/1 Knapsack

```
$ clingo-lpx encoding.lp  instance.lp --objective=global --quiet=1 0
clingo-lpx version 1.3.0
Reading from knapsack.lp
Solving...
Answer: 4
item(a) item(b) item(c) item(d)
load("9.1")
pack(a) pack(b)
value(a,"3.1") value(b,"3.2")
value(c,"1.9") value(d,"4.8")
weight(a,"3.3") weight(b,"4.7")
weight(c,"6.1") weight(d,"5.9")
Assignment:
a=1 b=1 c=0 d=0
Optimization: 63/10 [bounded]
SATISFIABLE

Models       : 4
Calls        : 1
Time         : 0.001s (Solving: 0.00s 1st Model: 0.00s Unsat: 0.00s)
CPU Time     : 0.001s
```

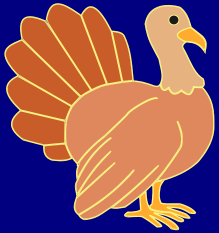


eclingo

- Idea extend *clingo* with subjective literals in the body of rules
- Features
  - subjective literal `&k{p}` holds if `p` holds in all stable models
  - natural representation of problems with elevated complexity (without saturation)
- References [31]
- Applications conformant planning [25] and action reversibility [55]

Example

Conformant planning

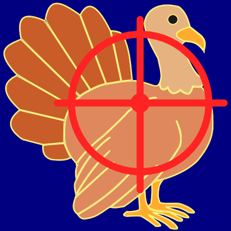


```
time(1..n). action(load). action(trigger).

{ holds(   dry,0); -holds(   dry,0) } = 1.
{ holds(loaded,0); -holds(loaded,0) } = 1.

{ occurs(A,T) : action(A) } = 1 :- time(T).

 holds(loaded,T) :- occurs(   load,T).
-holds(loaded,T) :- occurs(trigger,T).
-holds(   dry,T) :- occurs(trigger,T), holds(loaded,T-1).

 holds(F,T) :-  holds(F,T-1), not -holds(F,T), time(T).
-holds(F,T) :- -holds(F,T-1), not  holds(F,T), time(T).

goal :- -holds(dry,n).

% conformant planning

:- action(A), time(T), occurs(A,T), not &k{ occurs(A,T) }.
:- action(A), time(T), not occurs(A,T), &k{ occurs(A,T) }.

:- not &k { goal }.
```

Potassco

Example

Conformant planning

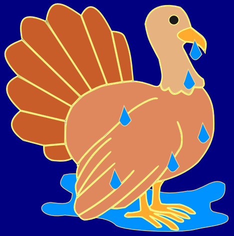


```
time(1..n). action(load). action(trigger).

{ holds(   dry,0); -holds(   dry,0) } = 1.
{ holds(loaded,0); -holds(loaded,0) } = 1.

{ occurs(A,T) : action(A) } = 1 :- time(T).

 holds(loaded,T) :- occurs(   load,T).
-holds(loaded,T) :- occurs(trigger,T).
-holds(   dry,T) :- occurs(trigger,T), holds(loaded,T-1).

 holds(F,T) :-  holds(F,T-1), not -holds(F,T), time(T).
-holds(F,T) :- -holds(F,T-1), not  holds(F,T), time(T).

goal :- -holds(dry,n).

% conformant planning

:- action(A), time(T), occurs(A,T), not &k{ occurs(A,T) }.
:- action(A), time(T), not occurs(A,T), &k{ occurs(A,T) }.

:- not &k { goal }.
```

Potassco

Example

Conformant planning



```
$ eclingo yale.lp -c n=2
eclingo version 0.2.0
Solving...
Answer: 1
&k{ occurs(load,1) }
&k{ occurs(trigger,2) }
&k{ goal }
SATISFIABLE

Elapsed time: 0.002736 s
```

Potassco

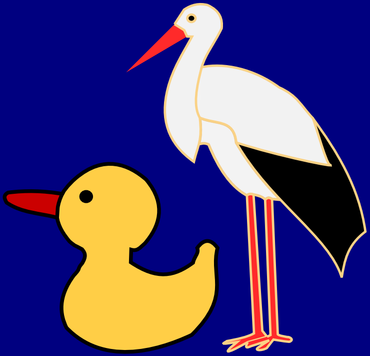
plingo

- Idea extends *clingo* with probabilistic reasoning
- Features
  - computes most probable model, exact or approximate queries
  - three frontends: *lpmln*, *p-log*, and *problog*
- Technology uses optimization to account for probabilities
- References [79]
- Applications incorporate uncertainty, like failure

Example

Birds in *lpmln*

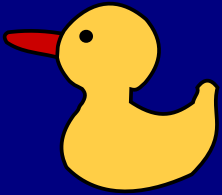
```
bird(X) :- resident(X).
bird(X) :- migratory(X).
:- resident(X), migratory(X).

resident(jo)  :- &weight(2).
migratory(jo) :- &weight(1).

&query(resident(jo)).
```



Example

Birds in *lpmln*

```
$ plingo birds.plp --frontend=lpmln-alt
plingo version 1.0.0
Reading from birds.plp
Solving...
Answer: 1

Optimization: 300000
Answer: 2
resident(jo) bird(jo)
Optimization: 100000
Answer: 3
migratory(jo) bird(jo)
Optimization: 200000

resident(jo): 0.66524

OPTIMUM FOUND

Models       : 3
  Optimum    : yes
Calls        : 1
Time         : 0.002s (Solving: 0.00s 1st Model: 0.00s Unsat: 0.00s)
CPU Time     : 0.002s
```



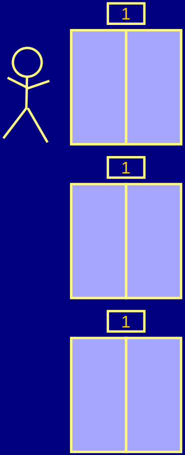
telingo

- Idea extend *clingo* with means for addressing dynamic problems
- Features
  - adds language elements from temporal, dynamic, (metric) logics eg.
    - next, always, eventually, until, etc
    - regular expressions
    - (intervals)
  - computes shortest trajectories
- References [33, 32]

Example

Elevator control

```
#program init.
called(3).
at(1).
floor(1..3).

#program always.
{ wait; up; down; serve } = 1 :- not &final.
:- up,   at(X), not _floor(X+1).
:- down, at(X), not _floor(X-1).

at(X+1):- 'up,   'at(X).
at(X-1):- 'down, 'at(X).
at(X)  :- 'at(X), not 'up, not 'down.

called(X):- 'called(X), #false:'at(X), 'serve.

:- called(X), &final.
ready :- called(X), at(X).

:- not &del{
  *( (*up + *down);; ?ready;; serve);; *wait .>? &final
  }, &initial.
```

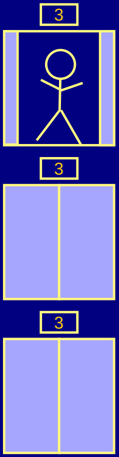


Potassco

Example

Elevator control

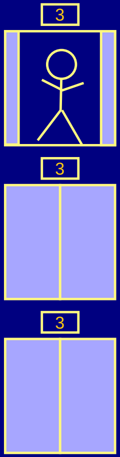
```
#program init.
called(3).
at(1).
floor(1..3).

#program always.
{ wait; up; down; serve } = 1 :- not &final.
:- up,   at(X), not _floor(X+1).
:- down, at(X), not _floor(X-1).

at(X+1):- 'up,   'at(X).
at(X-1):- 'down, 'at(X).
at(X)  :- 'at(X), not 'up, not 'down.

called(X):- 'called(X), #false:'at(X), 'serve.

:- called(X), &final.
ready :- called(X), at(X).

:- not &del{
  *( (*up + *down);; ?ready;; serve);; *wait .>? &final
  }, &initial.
```



Example

Elevator control

```
#program init.
called(3).
at(1).
floor(1..3).

#program always.
{ wait; up; down; serve } = 1 :- not &final.
:- up,   at(X), not _floor(X+1).
:- down, at(X), not _floor(X-1).

at(X+1):- 'up,   'at(X).
at(X-1):- 'down, 'at(X).
at(X)  :- 'at(X), not 'up, not 'down.

called(X):- 'called(X), #false:'at(X), 'serve.

:- called(X), &final.
ready :- called(X), at(X).

:- not &del{
  *( (*up + *down);; ?ready;; serve);; *wait .>? &final
  }, &initial.
```



Potassco

xclingo

- Idea extend *clingo* with explanations of answer sets
- Features
  - adds text annotations to display proofs
- References [27]
- Origin Made at Potassco Solutions Spain

Example

Blocks world

```
% !trace { occurs(move(B,From,To),T),                                      \
           "We moved block % from % to % at step %",B,From,To,T }   \
         :- occurs(move(B,From,To),T).

% !trace { holds(on(B),L,T),                                               \
           "Block % was % on %",B,Txt,L }                                  \
         :- holds(on(B),L,T), timetext(T,Txt).

% !trace { holds(on(B),L,T),                                               \
           "Block % was on % at %",B,L,T }                                 \
         :- holds(on(B),L,T), changed(on(B),T), T!=last, T!=0.

% !show_trace { holds(F,V,last) }.
```

```
*
|__"Block 14 was finally on 13"
|  |__"Block 14 was on 13 at 22"
|  |  |__"We moved block 14 from table to 13 at step 22"
|  |  |  |__"Block 14 was on table at 12"
|  |  |  |  |__"We moved block 14 from 11 to table at step 12"
|  |  |  |  |  |__"Block 14 was initially on 11"
```

Example

Blocks world

```
% !trace { occurs(move(B,From,To),T),                              \
           "We moved block % from % to % at step %",B,From,To,T } \
         :- occurs(move(B,From,To),T).

% !trace { holds(on(B),L,T),                                       \
           "Block % was % on %",B,Txt,L }                          \
         :- holds(on(B),L,T), timetext(T,Txt).

% !trace { holds(on(B),L,T),                                       \
           "Block % was on % at %",B,L,T }                         \
         :- holds(on(B),L,T), changed(on(B),T), T!=last, T!=0.

% !show_trace { holds(F,V,last) }.


*
|__"Block 14 was finally on 13"
|  |__"Block 14 was on 13 at 22"
|  |  |__"We moved block 14 from table to 13 at step 22"
|  |  |  |__"Block 14 was on table at 12"
|  |  |  |  |__"We moved block 14 from 11 to table at step 12"
|  |  |  |  |  |__"Block 14 was initially on 11"
```

Potassco

Outline

clinsight

- Idea improved development experience through ASP language support
- Features
  - real-time error diagnostics
  - detailed hover information for predicates
  - code completion
  - automatic safety checks for variables
- Technology Language server protocol, tree-sitter grammar
- Application development

clingo-server

- Idea run *clingo* as a server and use it via a Web API
- API methods
  - create a solver
  - register a theory
  - add logic programs
  - initiate grounding
  - set value of external atoms
  - initiate solving (with assumptions)
  - poll models
  - resume solving
  - finalize search
- Applications running ASP-based systems in the cloud

Example

Minimal usage

```
# Create solver
response = requests.get("http://my-clingo-server.de:8000/create")

# Upload logic program / facts
with open("queens.lp", "rb") as f:
    response = requests.post(
        "http://my-clingo-server.de:8000/add",
        data=f.read(),
        headers={"Content-Type": "text/plain; charset=utf-8 "},
    )

# Initiate grounding
response = requests.post(
    "http://my-clingo-server.de:8000/ground",
    data=io.StringIO('{"base": []}').read(),
    headers={"Content-Type": "application/json; charset=utf-8 "},
)

# Initiate solving
response = requests.get("http://my-clingo-server.de:8000/solve")

# Poll models
response = requests.get("http://my-clingo-server.de:8000/model", timeout=1)

if response.json() == "Running":
    print("No model yet ... ")
if "Model" in response.json():
    model = response.json()["Model"]
    print("Model found:")
    print(bytes(model).decode("utf-8"))
if response.json() == "Done":
    print("Search finished, no more models.")

# Close solve handle
response = requests.get("http://my-clingo-server.de:8000/close")
```

acclingo

- Idea find good *clingo* parameters for a set of instances
- Features
  - optimize runtime or solution quality
  - progressively selects better parameters via Bayesian optimization
  - built-in support for *clingo*
  - easily adaptable to *clingo*-based systems
- Technology SMAC3, a hyperparameter optimization tool

anthem

- Idea verify logic programs
- Features
  - translate logic programs to first-order theories
  - verify properties of the original programs, such as strong and external equivalence
  - different back-end theorem provers, eg *vampire*
- References [94, 56]

Example

Make your choice

```
% choice.1.lp
{q(X)} :- p(X).
```

```
% choice.2.lp
q(X) :- p(X), not not q(X).
```

```
$ anthem verify \
$     --equivalence strong \
$     choice.1.lp choice.2.lp
<snip>
> Success! Anthem found a proof of equivalence.
```

Potassco

asprin

- Idea optimize preferences among stable models
- Features
  - combinations of qualitative and quantitative preferences
  - easy implementation of new preferences
  - library of existing preference concepts
- References [23, 4, 22]

Example

Holiday preferences

```
#preference(costs,less(weight)){
  C :: sauna : cost(sauna,C);
  C ::  dive : cost(dive,C)
}.
#preference(fun,superset){
  sauna; dive; hike; not bunji
}.
#preference(temps,aso){
  dive >> sauna ||     hot;
 sauna >> dive  || not hot
}.
#preference(all,pareto){**costs; **fun; **temps}.

#optimize(all).
```

Potassco

asprin's library

- Basic preference types
  - `subset` and `superset`
  - `less(cardinality)` and `more(cardinality)`
  - `less(weight)` and `more(weight)`
  - `aso` (Answer Set Optimization)
  - `poset` (Qualitative Preferences)
  - etc.
- Composite preference types
  - `neg`
  - `and`
  - `pareto`
  - `lexico`
  - etc.

clingraph

- Idea build graph visualizations in ASP
- Features
  - graphs defined in terms of logic programs
  - uses *graphviz* as a backend
  - produces png, pdf, gif, (interactive) svg images and LaTeX code
- Technology *clingo*, *clorm*, *graphviz*
- References [78]
- Applications debugging, explanation, visualization

Potassco

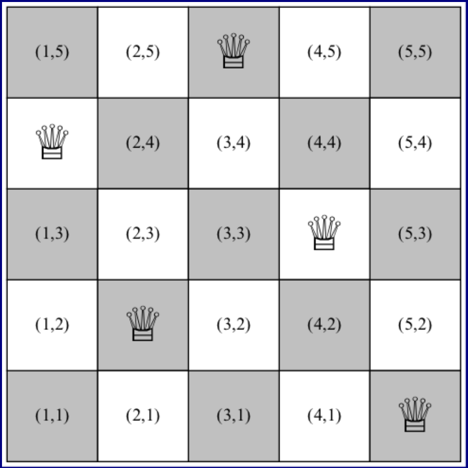
Example

Queens puzzle

```
node((X,Y)) :- cell(X,Y).

attr(node,(X,Y),width,1) :- cell(X,Y).
attr(node,(X,Y),pos,@pos(X,Y)) :- cell(X,Y).
attr(node,(X,Y),shape,square) :- cell(X,Y).
attr(node,(X,Y),style,filled) :- cell(X,Y).
attr(node,(X,Y),fillcolor,gray) :- cell(X,Y),(X+Y)\2=0.
attr(node,(X,Y),fillcolor,white) :- cell(X,Y),(X+Y)\2!=0.
attr(node,(X,Y),fontsize,"50") :- queen(X,Y).
attr(node,(X,Y),label,"♛") :- queen(X,Y).
```

Example

Queens puzzle



clinguin

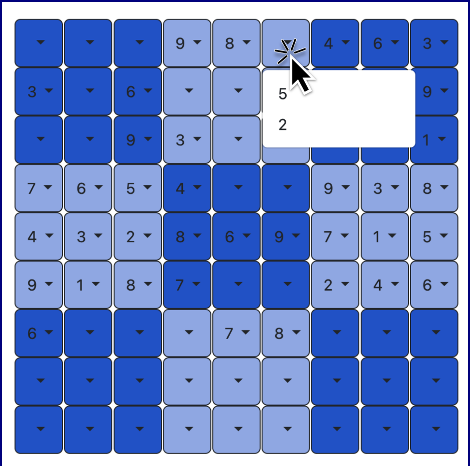
- Idea build user interfaces in ASP
- Features
  - UIs defined in terms of facts
  - extensible set of backend operations for interaction
  - brave reasoning for menus (via predicate `_any(X)`)
- Technology *clingo*, *clorm*, *angular*, *bootstrap*
- Applications product configuration, study regulations [80]

Example
Sudoku interactive

```
elem(window, window, root).
attr(window, child_layout, grid).

elem(d(X,Y), dropdown_menu, window) :- pos(X,Y).
attr(d(X,Y), width, 50)  :- pos(X,Y).
attr(d(X,Y), height, 50) :- pos(X,Y).
attr(d(X,Y), grid_column, X) :- pos(X,Y).
attr(d(X,Y), grid_row, Y) :- pos(X,Y).
attr(d(X,Y), class, ("border-dark";"bg-primary")) :- pos(X,Y).
attr(d(X,Y), class, "bg-opacity-50") :- subgrid(X,Y,S), S\2!=0.
attr(d(X,Y), selected, V) :- _all(sudoku(X,Y, V)).

elem(dv(X,Y, V), dropdown_menu_item, d(X,Y)) :- _any(sudoku(X,Y, V)).
attr(dv(X,Y, V), label, V) :- _any(sudoku(X,Y, V)).
when(dv(X,Y, V), click, call, add_assumption(sudoku(X,Y, V))) :- _any(sudoku(X,Y, V)).
```

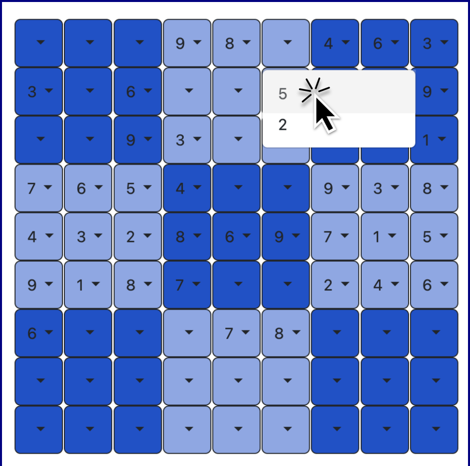
Example

Sudoku interactive



Example

Sudoku interactive



Potassco

Example

Sudoku interactive

			9	8	5	4	6	3
3	5	6	1	4	7	8	2	9
8	4	9	3	2	6	5	7	1
7	6	5	4	1	2	9	3	8
4	3	2	8	6	9	7	1	5
9	1	8	7	5	3	2	4	6
6				7	8			4

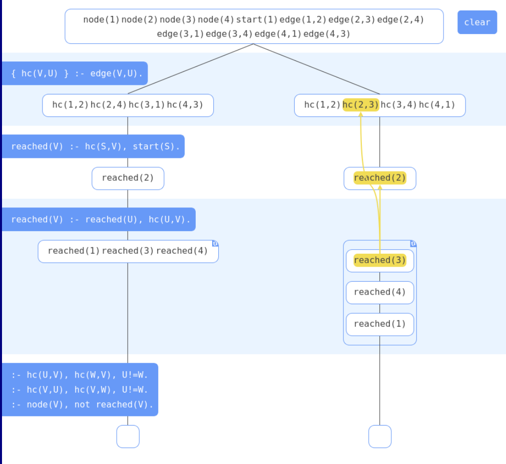
Viasp

- Idea visualize logic programs and their answer sets
- Features
  - fast and simple
  - for beginners
  - for (simple) explanation
  - web-browser frontend

Example

Hamiltonian cycle

```
$ viasp hamiltonian.lp 0
```



clintest

- Idea test logic programs
- Features
  - numerous off-the-shelf tests
  - support for integrating custom-build tests
  - efficent test execution by steering the solving process
  - support for optimization
- Technology *clingo*

Example

Testing at work

```
from clintest.test import Assert, And
from clintest.quantifier import All, Any
from clintest.assertion import Contains
from clintest.solver import Clingo

solver = Clingo("0", "a. {b}.")
test = And(
    Assert(Any(), Contains("a")),
    Assert(All(), Contains("b")),
    Assert(Any(), Contains("c")),
)

solver.solve(test)
test.assert_()
```

clorm

- Idea
  - simplify working with ASP facts via a Python library
  - simplify Python-*clingo* integration for easy refactoring and debugging
- Features
  - Python classes to define the mapping to *clingo* predicates
  - functions to import/export *clingo* solver facts
  - specialized container class with an intuitive query language
- Technology *clingo* (Python API)
- Origin Made at Potassco Solutions Australia

Example

- Given ASP facts with a common predicate signature:

```
assign("Bob", task1).    assign("Bob", task3).
assign("Bill", task2).
```

- Use a Python class to specify the mapping:

```
class Assign(Predicate):
   person: str
   task: ConstantStr
```

- Then easily query a set of facts stored in a *FactBase* container:

```
query = factbase.query(Assign) \
                .where(Assign.person == "Bob") \
                .order_by(Assign.task)
for assign in query.all():
    print(assign)
```

ngo

- Idea improve non-ground logic programs
- Features
  - instance independent
  - detects symmetries
  - improves aggregate translation
  - improves optimization statements
  - improves mathematical computations
  - removes unnecessary variables/predicates
- Slogan Write clean ASP code without worrying about performance!

Outline

Application-oriented systems

- *aspartame* constraint solver
- *aspcafe* vehicle equipment specification
- *aspcud* software package configuration
- *asprilo* warehouse simulation
- *chasp* music composition
- *flatzingo* constraint solver
- *fluto* metabolic network expansion
- *plasp* planning system
- *qasp* quantified ASP solver
- *spa* study planner
- *teaspoon* university timetabling system
- *xorro* sampling stable models

Outline

Summary

- Meta programming allows for rapid prototyping of ASP extensions
- True applications and extensions rest upon the API of *clingo*

Auf Wiedersehen!

"Tomorrow isn't staying out
I'll be back, without a doubt!"

Pink panther

Bibliography

- The following list of references is compiled from the open source bibliography available at

  `https://github.com/krr-up/bibliography`

- Feel free to submit corrections via pull requests !

[1] D. Abels et al. “Train scheduling with hybrid ASP”. In: *Theory and Practice of Logic Programming* 21.3 (2021), pp. 317–347. DOI: 10.1017/S1471068420000046.

[2] S. Abiteboul, R. Hull, and V. Vianu. *Foundations of Databases*. Addison-Wesley, 1995.

[3] I. Aini, A. Saptawijaya, and S. Aminah. “Bringing answer set programming to the next level: A real case on modeling course timetabling”. In: *International Conference on Advanced Computer Science and Information Systems*. Institute of Electrical and Electronics Engineers Inc., May 2018, pp. 471–476. DOI: 10.1109/ICACSIS.2017.aisaam18a.

[4] M. Alviano, J. Romero, and T. Schaub. “Preference Relations by Approximation”. In: *Proceedings of the Sixteenth International Conference on Principles of Knowledge Representation and Reasoning (KR’18)*. Ed. by M. Thielscher, F. Toni, and F. Wolter. AAAI Press, 2018, pp. 2–11.

Potassco

[5] M. Alviano et al. “The ASP System DLV2”. In: *Proceedings of the Fourteenth International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR’17)*. Ed. by M. Balduccini and T. Janhunen. Vol. 10377. Lecture Notes in Artificial Intelligence. Springer-Verlag, 2017, pp. 215–221.

[6] M. Alviano et al. “The Fourth Answer Set Programming Competition: Preliminary Report”. In: *Proceedings of the Twelfth International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR’13)*. Ed. by P. Cabalar and T. Son. Vol. 8148. Lecture Notes in Artificial Intelligence. Springer-Verlag, 2013, pp. 42–53.

[7] M. Alviano et al. “WASP: A Native ASP Solver Based on Constraint Learning”. In: *Proceedings of the Twelfth International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR’13)*. Ed. by P. Cabalar and T. Son. Vol. 8148. Lecture Notes in Artificial Intelligence. Springer-Verlag, 2013, pp. 54–66.

[8] J. Babb and J. Lee. "Cplus2ASP: Computing Action Language C+ in Answer Set Programming". In: *Proceedings of the Twelfth International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR'13)*. Ed. by P. Cabalar and T. Son. Vol. 8148. Lecture Notes in Artificial Intelligence. Springer-Verlag, 2013, pp. 122–134.

[9] M. Balduccini, Y. Lierler, and S. Woltran, eds. *Proceedings of the Fifteenth International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR'19)*. Vol. 11481. Lecture Notes in Artificial Intelligence. Springer-Verlag, 2019.

[10] M. Banbara et al. "Clingcon: The Next Generation". In: *Theory and Practice of Logic Programming* 17.4 (2017), pp. 408–461. DOI: 10.1017/S1471068417000138.

[11] M. Banbara et al. "teaspoon: Solving the Curriculum-Based Course Timetabling Problems with Answer Set Programming". In: *Annals of Operations Research* 275.1 (2019), pp. 3–37. DOI: 10.1007/s10479-018-2757-7.

[12] C. Baral. *Knowledge Representation, Reasoning and Declarative Problem Solving*. Cambridge University Press, 2003.

[13] C. Baral, G. Brewka, and J. Schlipf, eds. *Proceedings of the Ninth International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR'07)*. Vol. 4483. Lecture Notes in Artificial Intelligence. Springer-Verlag, 2007.

[14] C. Baral and M. Gelfond. "Logic Programming and Knowledge Representation". In: *Journal of Logic Programming* 19/20 (1994), pp. 73–148.

[15] R. Bertolucci et al. "Manipulation of Articulated Objects using Dual-arm Robots via Answer Set Programming". In: *Theory and Practice of Logic Programming* 21.3 (2021), pp. 372–401. DOI: 10.1017/S1471068420000459.

[16] A. Bogatarkan and E. Erdem. "Explanation Generation for Multi-Modal Multi-Agent Path Finding with Optimal Resource Utilization using Answer Set Programming". In: *Theory and Practice of Logic Programming* 20.6 (2020), pp. 974–989. DOI:

10.1017/S1471068420000320. URL: https://doi.org/10.1017/S1471068420000320.

[17] J. Bomanson, T. Janhunen, and A. Weinzierl. "Enhancing Lazy Grounding with Lazy Normalization in Answer-Set Programming". In: *Proceedings of the Thirty-third National Conference on Artificial Intelligence (AAAI'19)*. Ed. by P. Van Hentenryck and Z. Zhou. AAAI Press, 2019, pp. 2694–2702.

[18] P. Borchert et al. "Towards Systematic Benchmarking in Answer Set Programming: The Dagstuhl Initiative". In: *Proceedings of the Seventh International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR'04)*. Ed. by V. Lifschitz and I. Niemelä. Vol. 2923. Lecture Notes in Artificial Intelligence. Springer-Verlag, 2004, pp. 3–7.

[19] G. Brewka, T. Eiter, and M. Truszczynski. "Answer Set Programming: An Introduction to the Special Issue". In: *AI Magazine* 37.3 (2016), pp. 5–6.

[20] G. Brewka, T. Eiter, and M. Truszczyński. "Answer set programming at a glance". In: *Communications of the ACM* 54.12 (2011), pp. 92–103. DOI: 10.1145/2043174.2043195.

[21] G. Brewka, I. Niemelä, and M. Truszczyński. "Answer Set Optimization". In: *Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence (IJCAI'03)*. Ed. by G. Gottlob and T. Walsh. Morgan Kaufmann Publishers, 2003, pp. 867–872.

[22] G. Brewka et al. "A general framework for preferences in answer set programming". In: *Artificial Intelligence* (2023). To appear.

[23] G. Brewka et al. "asprin: Customizing Answer Set Preferences without a Headache". In: *Proceedings of the Twenty-ninth National Conference on Artificial Intelligence (AAAI'15)*. Ed. by B. Bonet and S. Koenig. AAAI Press, 2015, pp. 1467–1474. URL: http://www.aaai.org/ocs/index.php/AAAI/AAAI15/paper/view/9535.

[24] G. Brewka et al. "Implementing Preferences with asprin". In: *Proceedings of the Thirteenth International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR'15)*. Ed. by F. Calimeri, G. Ianni, and M. Truszczyński. Vol. 9345. Lecture Notes in Artificial Intelligence. Springer-Verlag, 2015, pp. 158–172.

[25] P. Cabalar, J. Fandinno, and L. Fariñas del Cerro. "Splitting Epistemic Logic Programs". In: *Theory and Practice of Logic Programming* 21 (2021), pp. 296–316.

[26] P. Cabalar, J. Fandinno, and B. Muñiz. "A System for Explainable Answer Set Programming". In: *Proceedings of the International Conference on Logic Programming*. Ed. by F. Ricca et al. Vol. 325. Electronic Proceedings in Theoretical Computer Science. 2020, pp. 124–136. DOI: 10.4204/EPTCS.325.19.

[27] P. Cabalar, J. Fandinno, and B. Muñiz. "A System for Explainable Answer Set Programming". In: *Technical Communications of the Thirty sixth International Conference on Logic Programming*

(ICLP'20). Ed. by F. Ricca et al. Vol. 325. EPTCS. 2020, pp. 124–136. DOI: 10.4204/EPTCS.325.19.

[28] P. Cabalar and T. Son, eds. *Proceedings of the Twelfth International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR'13)*. Vol. 8148. Lecture Notes in Artificial Intelligence. Springer-Verlag, 2013.

[29] P. Cabalar et al. "A Uniform Treatment of Aggregates and Constraints in Hybrid ASP". In: *Proceedings of the Seventeenth International Conference on Principles of Knowledge Representation and Reasoning (KR'21)*. Ed. by D. Calvanese, E. Erdem, and M. Thielscher. AAAI Press, 2020, pp. 193–202.

[30] P. Cabalar et al. "An ASP semantics for Constraints involving Conditional Aggregates". In: *Proceedings of the Twenty-fourth European Conference on Artificial Intelligence (ECAI'20)*. Ed. by G. De Giacomo et al. IOS Press, 2020, pp. 664–671.

[31] P. Cabalar et al. "eclingo: A solver for Epistemic Logic Programs". In: *Theory and Practice of Logic Programming* 20.5 (2020). `https://github.com/potassco/eclingo`, pp. 834–847.

[32] P. Cabalar et al. "Implementing Dynamic Answer Set Programming over finite traces". In: *Proceedings of the Twenty-fourth European Conference on Artificial Intelligence (ECAI'20)*. Ed. by G. De Giacomo et al. IOS Press, 2020, pp. 656–663. DOI: `10.3233/FAIA200151`.

[33] P. Cabalar et al. "telingo = ASP + Time". In: *Proceedings of the Fifteenth International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR'19)*. Ed. by M. Balduccini, Y. Lierler, and S. Woltran. Vol. 11481. Lecture Notes in Artificial Intelligence. Springer-Verlag, 2019, pp. 256–269. DOI: `10.1007/978-3-030-20528-7\_19`.

[34] F. Calimeri et al. "ASP-Core-2 Input Language Format". In: *Theory and Practice of Logic Programming* 20.2 (2020), pp. 294–309.

Potassco

[35] F. Calimeri et al. "ASP-Core-2 Input Language Format". In: *Theory and Practice of Logic Programming* 20.2 (2020), pp. 294–309.

[36] F. Calimeri et al. "I-DLV: The new intelligent grounder of DLV". In: *Intelligenza Artificiale* 11.1 (2017), pp. 5–20.

[37] F. Calimeri et al. "The Design of the Fifth Answer Set Programming Competition". In: *Technical Communications of the Thirtieth International Conference on Logic Programming (ICLP'14)*. Ed. by M. Leuschel and T. Schrijvers. Vol. arXiv:1405.3710v4. Theory and Practice of Logic Programming, Online Supplement. 2014. URL: `http://arxiv.org/abs/1405.3710v4`.

[38] F. Calimeri et al. "The Third Answer Set Programming Competition: Preliminary Report of the System Competition Track". In: *Proceedings of the Eleventh International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR'11)*. Ed. by J. Delgrande and W. Faber. Vol. 6645. Lecture Notes in Artificial Intelligence. Springer-Verlag, 2011, pp. 388–403.

[39] D. Calvanese, E. Erdem, and M. Thielscher, eds. *Proceedings of the Seventeenth International Conference on Principles of Knowledge Representation and Reasoning (KR'21)*. AAAI Press, 2020.

[40] K. Compton, A. Smith, and M. Mateas. "Anza Island: Novel Gameplay Using ASP". In: *Proceedings of the The third workshop on Procedural Content Generation in Games*. ACM Press, 2012, 13:1–13:4. DOI: 10.1145/2538528.2538539. URL: https://doi.org/10.1145/2538528.2538539.

[41] M. D'Agostino et al., eds. *Handbook of Tableau Methods*. Kluwer Academic Publishers, 1999.

[42] C. Dabral and C. Martens. "Generating Explorable Narrative Spaces with Answer Set Programming". In: *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment* 16.1 (Oct. 2020), pp. 45–51. URL: https://ojs.aaai.org/index.php/AIIDE/article/view/7406.

[43] G. De Giacomo et al., eds. *Proceedings of the Twenty-fourth European Conference on Artificial Intelligence (ECAI'20)*. IOS Press, 2020.

[44] J. Delgrande and W. Faber, eds. *Proceedings of the Eleventh International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR'11)*. Vol. 6645. Lecture Notes in Artificial Intelligence. Springer-Verlag, 2011.

[45] M. Denecker et al. "The Second Answer Set Programming Competition". In: *Proceedings of the Tenth International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR'09)*. Ed. by E. Erdem, F. Lin, and T. Schaub. Vol. 5753. Lecture Notes in Artificial Intelligence. Springer-Verlag, 2009, pp. 637–654.

[46] E. Di Rosa, E. Giunchiglia, and M. Maratea. "Solving satisfiability problems with preferences". In: *Constraints* 15.4 (2010), pp. 485–515.

[47] Y. Dimopoulos et al. "Encoding Reversing Petri Nets in Answer Set Programming". In: *Reversible Computation - 12th International Conference*. Ed. by I. Lanese and M. Rawski. Vol. 12227. Lecture Notes in Computer Science. Springer-Verlag, 2020, pp. 264–271. DOI: 10.1007/978-3-030-52482-1_17. URL: https://doi.org/10.1007/978-3-030-52482-1%5C_17.

[48] Y. Dimopoulos et al. "plasp 3: Towards Effective ASP Planning". In: *Theory and Practice of Logic Programming* 19.3 (2018), pp. 477–504.

[49] W. Dvorák et al. "ASPARTIX-V19 - An Answer-Set Programming Based System for Abstract Argumentation". In: *Foundations of Information and Knowledge Systems - 11th International Symposium*. Ed. by A. Herzig and J. Kontinen. Vol. 12012. Lecture Notes in Computer Science. Springer-Verlag, 2020, pp. 79–89. DOI: 10.1007/978-3-030-39951-1_5. URL: https://doi.org/10.1007/978-3-030-39951-1%5C_5.

[50] T. Eiter, G. Ianni, and T. Krennwallner. "Answer Set Programming: A Primer". In: *Fifth International Reasoning Web Summer School (RW'09)*. Ed. by S. Tessaris et al. Vol. 5689. Lecture Notes in Computer Science. Slides at http://www.kr.tuwien.ac.at/staff/tkren/pub/2009/rw2009-lecture.zip. Springer-Verlag, 2009, pp. 40–110. URL: `http://www.kr.tuwien.ac.at/staff/tkren/pub/2009/rw2009-asp.pdf`.

[51] T. Eiter et al. "Answer Set Programming with External Source Access". In: *Reasoning Web. Semantic Interoperability on the Web - 13th International Summer School 2017*. Ed. by G. Ianni et al. Vol. 10370. Lecture Notes in Computer Science. Springer-Verlag, 2017, pp. 204–275. DOI: `10.1007/978-3-319-61033-7\_7`. URL: `https://doi.org/10.1007/978-3-319-61033-7%5C_7`.

[52] E. Erdem and A. Herzig. "Solving Gossip Problems using Answer Set Programming: An Epistemic Planning Approach". In: *Proceedings of the International Conference on Logic Programming*. Ed. by F. Ricca et al. Vol. 325. Electronic

Proceedings in Theoretical Computer Science. 2020, pp. 52–58. DOI: 10.4204/EPTCS.325.11. URL: https://doi.org/10.4204/EPTCS.325.11.

[53] E. Erdem, F. Lin, and T. Schaub, eds. *Proceedings of the Tenth International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR'09)*. Vol. 5753. Lecture Notes in Artificial Intelligence. Springer-Verlag, 2009.

[54] E. Erdem et al. "A General Framework for Stable Roommates Problems using Answer Set Programming". In: *Theory and Practice of Logic Programming* 20.6 (2020), pp. 911–925. DOI: 10.1017/S1471068420000277. URL: https://doi.org/10.1017/S1471068420000277.

[55] W. Faber, M. Morak, and L. Chrpa. "Determining Action Reversibility in STRIPS Using Answer Set and Epistemic Logic Programming". In: *Theory and Practice of Logic Programming* 21.5 (2021), pp. 646–662.

Potassco

[56] J. Fandinno et al. "Verifying Tight Logic Programs with anthem and vampire". In: *Theory and Practice of Logic Programming* 20.5 (2020), pp. 735–750.

[57] M. Fitting. "A Kripke-Kleene Semantics for Logic Programs". In: *Journal of Logic Programming* 2.4 (1985), pp. 295–312.

[58] M. Fitting. "Fixpoint semantics for logic programming: A survey". In: *Theoretical Computer Science* 278.1-2 (2002), pp. 25–51.

[59] C. Frioux et al. "Hybrid Metabolic Network Completion". In: *Theory and Practice of Logic Programming* 19.1 (2019), pp. 83–108.

[60] S. Gaggl et al. "Improved answer-set programming encodings for abstract argumentation". In: *Theory and Practice of Logic Programming* 15.4-5 (2015), pp. 434–448. DOI: 10.1017/S1471068415000149. URL: https://doi.org/10.1017/S1471068415000149.

[61] M. Gebser, R. Kaminski, and T. Schaub. "Complex Optimization in Answer Set Programming". In: *Theory and Practice of Logic Programming* 11.4-5 (2011), pp. 821–839.

[62] M. Gebser, B. Kaufmann, and T. Schaub. "Conflict-Driven Answer Set Solving: From Theory to Practice". In: *Artificial Intelligence* 187-188 (2012), pp. 52–89. DOI: 10.1016/j.artint.2012.04.001.

[63] M. Gebser, B. Kaufmann, and T. Schaub. "Multi-threaded ASP Solving with clasp". In: *Theory and Practice of Logic Programming* 12.4-5 (2012), pp. 525–545.

[64] M. Gebser, T. Schaub, and S. Thiele. "Gringo: A New Grounder for Answer Set Programming". In: *Proceedings of the Ninth International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR'07)*. Ed. by C. Baral, G. Brewka, and J. Schlipf. Vol. 4483. Lecture Notes in Artificial Intelligence. Springer-Verlag, 2007, pp. 266–271.

[65] M. Gebser et al. "Abstract Gringo". In: *Theory and Practice of Logic Programming* 15.4-5 (2015), pp. 449–463. DOI: 10.1017/S1471068415000150.

[66] M. Gebser et al. "Advances in gringo Series 3". In: *Proceedings of the Eleventh International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR'11)*. Ed. by J. Delgrande and W. Faber. Vol. 6645. Lecture Notes in Artificial Intelligence. Springer-Verlag, 2011, pp. 345–351.

[67] M. Gebser et al. *Answer Set Solving in Practice*. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan and Claypool Publishers, 2012. DOI: 10.1007/978-3-031-01561-8.

[68] M. Gebser et al. "Conflict-Driven Answer Set Solving". In: *Proceedings of the Twentieth International Joint Conference on Artificial Intelligence (IJCAI'07)*. Ed. by M. Veloso. AAAI/MIT Press, 2007, pp. 386–392.

[69] M. Gebser et al. "Multi-shot ASP solving with clingo". In: *Theory and Practice of Logic Programming* 19.1 (2019), pp. 27–82. DOI: 10.1017/S1471068418000054.

[70] M. Gebser et al. "On the Input Language of ASP Grounder Gringo". In: *Proceedings of the Tenth International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR'09)*. Ed. by E. Erdem, F. Lin, and T. Schaub. Vol. 5753. Lecture Notes in Artificial Intelligence. Springer-Verlag, 2009, pp. 502–508.

[71] M. Gebser et al. *Potassco User Guide*. 2nd ed. University of Potsdam. 2015. URL: http://potassco.org.

[72] M. Gebser et al. "The First Answer Set Programming System Competition". In: *Proceedings of the Ninth International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR'07)*. Ed. by C. Baral, G. Brewka, and J. Schlipf. Vol. 4483. Lecture Notes in Artificial Intelligence. Springer-Verlag, 2007, pp. 3–17.

[73] M. Gelfond. "Answer Sets". In: *Handbook of Knowledge Representation*. Ed. by V. Lifschitz, F. van Harmelen, and B. Porter. Elsevier Science, 2008. Chap. 7, pp. 285–316.

[74] M. Gelfond and Y. Kahl. *Knowledge Representation, Reasoning, and the Design of Intelligent Agents: The Answer-Set Programming Approach*. Cambridge University Press, 2014.

[75] M. Gelfond and N. Leone. "Logic programming and knowledge representation — the A-Prolog perspective". In: *Artificial Intelligence* 138.1-2 (2002), pp. 3–38.

[76] M. Gelfond and V. Lifschitz. "Logic Programs with Classical Negation". In: *Proceedings of the Seventh International Conference on Logic Programming (ICLP'90)*. Ed. by D. Warren and P. Szeredi. MIT Press, 1990, pp. 579–597.

[77] M. Gelfond and V. Lifschitz. "The Stable Model Semantics for Logic Programming". In: *Proceedings of the Fifth International Conference and Symposium of Logic Programming (ICLP'88)*.

Ed. by R. Kowalski and K. Bowen. MIT Press, 1988, pp. 1070–1080. DOI: 10.1201/b10397-6.

[78] S. Hahn et al. "clingraph: ASP-based Visualization". In: *Proceedings of the Sixteenth International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR'22)*. Ed. by G. Gottlob, D. Inclezan, and M. Maratea. Vol. 13416. Lecture Notes in Artificial Intelligence. Springer-Verlag, 2022, pp. 401–414. DOI: 10.1007/978-3-031-15707-3_31.

[79] S. Hahn et al. "Plingo: A System for Probabilistic Reasoning in Clingo Based on LP^{MLN}". In: *Proceedings of the Sixth International Joint Conference on Rules and Reasoning (RuleML+RR'22)*. Ed. by Guido Governatori and Anni-Yasmin Turhan. Vol. 13752. Lecture Notes in Computer Science. Springer-Verlag, 2022, pp. 54–62. DOI: 10.1007/978-3-031-21541-4_4.

[80] S. Hahn et al. "Reasoning about Study Regulations in Answer Set Programming (Preliminary Report)". In: *Proceedings of the International Conference on Logic Programming Workshops*. Ed. by

J. Arias et al. Vol. 3437. CEUR Workshop Proceedings. CEUR-WS.org, 2023. URL: https://ceur-ws.org/Vol-3437/paper5ASPOCP.pdf.

[81] C. Haubelt et al. "Evolutionary System Design with Answer Set Programming". In: *Algorithms* 16.4 (2023). ISSN: 1999-4893. DOI: 10.3390/a16040179. URL: https://www.mdpi.com/1999-4893/16/4/179.

[82] Y. Izmirlioglu and E. Erdem. "Reasoning about Cardinal Directions between 3-Dimensional Extended Objects using Answer Set Programming". In: *Theory and Practice of Logic Programming* 20.6 (2020), pp. 942–957. DOI: 10.1017/S1471068420000411. URL: https://doi.org/10.1017/S1471068420000411.

[83] T. Janhunen et al. "Clingo goes Linear Constraints over Reals and Integers". In: *Theory and Practice of Logic Programming* 17.5-6 (2017), pp. 872–888. DOI: 10.1017/S1471068417000242.

[84] R. Kaminski and T. Schaub. "On the Foundations of Grounding in Answer Set Programming". In: *Theory and Practice of Logic*

Programming (2023), pp. 1–60. DOI: 10.1017/S1471068422000308.

[85] R. Kaminski, T. Schaub, and P. Wanko. "A Tutorial on Hybrid Answer Set Solving with clingo". In: *Proceedings of the Thirteenth International Summer School of the Reasoning Web*. Ed. by G. Ianni et al. Vol. 10370. Lecture Notes in Computer Science. Springer-Verlag, 2017, pp. 167–203. DOI: 10.1007/978-3-319-61033-7_6.

[86] R. Kaminski et al. "How to Build Your Own ASP-based System?!" In: *Theory and Practice of Logic Programming* 23.1 (2023), pp. 299–361. DOI: 10.1017/S1471068421000508.

[87] J. Lee. "A Model-Theoretic Counterpart of Loop Formulas". In: *Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence (IJCAI'05)*. Ed. by L. Kaelbling and A. Saffiotti. Professional Book Center, 2005, pp. 503–508.

[88] N. Leone et al. "The DLV System for Knowledge Representation and Reasoning". In: *ACM Transactions on Computational Logic* 7.3 (2006), pp. 499–562.

[89] V. Lifschitz. *Answer Set Programming*. Springer-Verlag, 2019. DOI: 10.1007/978-3-030-24658-7.

[90] V. Lifschitz. "Answer set programming and plan generation". In: *Artificial Intelligence* 138.1-2 (2002), pp. 39–54.

[91] V. Lifschitz. "Introduction to answer set programming". Unpublished draft. 2004. URL: http://www.cs.utexas.edu/users/vl/papers/esslli.ps.

[92] V. Lifschitz. "Thirteen Definitions of a Stable Model". In: *Fields of Logic and Computation, Essays Dedicated to Yuri Gurevich on the Occasion of His 70th Birthday*. Ed. by A. Blass, N. Dershowitz, and W. Reisig. Vol. 6300. Lecture Notes in Computer Science. Springer-Verlag, 2010, pp. 488–503. DOI: 10.1007/978-3-642-15025-8_24.

Potassco

[93] V. Lifschitz. "Twelve Definitions of a Stable Model". In: *Proceedings of the Twenty-fourth International Conference on Logic Programming (ICLP'08)*. Ed. by M. Garcia de la Banda and E. Pontelli. Vol. 5366. Lecture Notes in Computer Science. Springer-Verlag, 2008, pp. 37–51.

[94] V. Lifschitz, P. Lühne, and T. Schaub. "Verifying Strong Equivalence of Programs in the Input Language of GRINGO". In: *Proceedings of the Fifteenth International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR'19)*. Ed. by M. Balduccini, Y. Lierler, and S. Woltran. Vol. 11481. Lecture Notes in Artificial Intelligence. Springer-Verlag, 2019, pp. 270–283.

[95] X. Liu and M. Truszczynski. "Aggregating Conditionally Lexicographic Preferences Using Answer Set Programming Solvers". In: *Algorithmic Decision Theory - Third International Conference*. Ed. by P. Perny, M. Pirlot, and A. Tsoukiàs. Vol. 8176. Lecture Notes in Computer Science. Springer-Verlag, 2013, pp. 244–258. DOI: 10.1007/978-3-642-41575-3_19. URL: https://doi.org/10.1007/978-3-642-41575-3_19.

[96] V. Marek and M. Truszczyński. *Nonmonotonic logic: context-dependent reasoning*. Artifical Intelligence. Springer-Verlag, 1993.

[97] V. Marek and M. Truszczyński. "Stable models and an alternative logic programming paradigm". In: *The Logic Programming Paradigm: a 25-Year Perspective*. Ed. by K. Apt et al. Springer-Verlag, 1999, pp. 375–398.

[98] N. Newman, A. Fréchette, and K. Leyton-Brown. "Deep optimization for spectrum repacking". In: *Communications of the ACM* 61.1 (2018), pp. 97–104.

[99] V. Nguyen et al. "Explainable Planning Using Answer Set Programming". In: *Proceedings of the Seventeenth International Conference on Principles of Knowledge Representation and Reasoning (KR'21)*. Ed. by D. Calvanese, E. Erdem, and M. Thielscher. AAAI Press, 2020, pp. 662–666.

[100] I. Niemelä. "Logic Programs with Stable Model Semantics as a Constraint Programming Paradigm". In: *Annals of Mathematics and Artificial Intelligence* 25.3-4 (1999), pp. 241–273.

[101] I. Niemelä and P. Simons. "Efficient Implementation of the Well-founded and Stable Model Semantics". In: *Proceedings of the Joint International Conference and Symposium on Logic Programming*. Ed. by M. Maher. MIT Press, 1996, pp. 289–303.

[102] M. Ostrowski and T. Schaub. "ASP modulo CSP: The clingcon system". In: *Theory and Practice of Logic Programming* 12.4-5 (2012), pp. 485–503. DOI: 10.1017/S1471068412000142.

[103] D. Rajaratnam et al. "Solving an Industrial-Scale Warehouse Delivery Problem with Answer Set Programming Modulo Difference Constraints". In: *Algorithms* 16.4 (2023). DOI: 10.3390/a16040216. URL: https://www.mdpi.com/1999-4893/16/4/216.

[104] M. Rezvani et al. "Analyzing XACML policies using answer set programming". In: *International Journal of Information Security*

18.4 (2019), pp. 465–479. DOI: 10.1007/s10207-018-0421-5. URL: https://doi.org/10.1007/s10207-018-0421-5.

[105] D. Saccá and C. Zaniolo. "Stable Models and Non-Determinism in Logic Programs with Negation". In: *Proceedings of the Ninth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*. ACM Press, 1990, pp. 205–217.

[106] T. Schaub and S. Woltran. "Answer set programming unleashed!" In: *Künstliche Intelligenz* 32.2-3 (2018), pp. 105–108.

[107] T. Schaub and S. Woltran. "Special Issue on Answer Set Programming". In: *Künstliche Intelligenz* 32.2-3 (2018), pp. 101–103.

[108] C. Schultz et al. "Answer Set Programming Modulo 'Space-Time'". In: *Rules and Reasoning - Second International Joint Conference*. Ed. by C. Benzmüller et al. Vol. 11092. Lecture Notes in Computer Science. Springer-Verlag, 2018, pp. 318–326. DOI: 10.1007/978-3-319-99906-7_24. URL: https://doi.org/10.1007/978-3-319-99906-7%5C_24.

Potassco

[109] R. Schwitter. "Answer Set Programming via Controlled Natural Language Processing". In: *Controlled Natural Language - Third International Workshop*. Ed. by T. Kuhn and N. Fuchs. Vol. 7427. Lecture Notes in Computer Science. Springer-Verlag, 2012, pp. 26–43. DOI: 10.1007/978-3-642-32612-7_3. URL: https://doi.org/10.1007/978-3-642-32612-7%5C_3.

[110] P. Simons, I. Niemelä, and T. Soininen. "Extending and implementing the stable model semantics". In: *Artificial Intelligence* 138.1-2 (2002), pp. 181–234.

[111] A. Smith et al. "A case study of expressively constrainable level design automation tools for a puzzle game". In: *International Conference on the Foundations of Digital Games*. Ed. by M. El-Nasr, M. Consalvo, and S. Feiner. ACM Press, 2012, pp. 156–163. DOI: 10.1145/2282338.2282370. URL: https://doi.org/10.1145/2282338.2282370.

[112] T. Son and E. Pontelli. "Planning with Preferences using Logic Programming". In: *Theory and Practice of Logic Programming* 6.5 (2006), pp. 559–608.

[113] T. Syrjänen. "Omega-Restricted Logic Programs". In: *Proceedings of the Sixth International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR'01)*. Ed. by T. Eiter, W. Faber, and M. Truszczyński. Vol. 2173. Lecture Notes in Computer Science. Springer-Verlag, 2001, pp. 267–279.

[114] E. Teppan. and G. Friedrich. *Heuristic Constraint Answer Set Programming for Manufacturing Problems*. Ed. by I. Hatzilygeroudis and V. Palade. Springer-Verlag, 2018, pp. 119–147. ISBN: 978-3-319-66790-4. DOI: 10.1007/978-3-319-66790-4_7. URL: https://doi.org/10.1007/978-3-319-66790-4_7.

[115] M. Truszczyński. "An introduction to the stable and well-founded semantics of logic programs". In: *Declarative Logic Programming:*

Potassco

Theory, Systems, and Applications. Ed. by M. Kifer and Y. Liu. ACM / Morgan & Claypool, 2018, pp. 121–177.

[116] A. Van Gelder, K. Ross, and J. Schlipf. "The Well-Founded Semantics for General Logic Programs". In: *Journal of the ACM* 38.3 (1991), pp. 620–650.

[117] F. Wotawa. "On the Use of Answer Set Programming for Model-Based Diagnosis". In: *Proceedings of the Thirty-third International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems (IEA/AIE'20)*. Ed. by H. Fujita et al. Vol. 12144. Lecture Notes in Computer Science. Springer-Verlag, 2020, pp. 518–529.