Implementing preferences with *asprin*

Gerhard Brewka James Delgrande Javier Romero Torsten Schaub

University of Leipzig Simon Fraser University University of Potsdam INRIA Rennes

Outline

Outline

Potassco

Motivation

- The identification of preferred, or optimal, solutions is often indispensable in real-world applications

  In many cases, this also involves the combination of various qualitative and quantitative preferences

- Only optimization statements representing objective functions using summation are established components of today's ASP systems

- Example $\#minimize\{40 : sauna, 70 : dive\}$

Motivation

- The identification of preferred, or optimal, solutions is often indispensable in real-world applications

  In many cases, this also involves the combination of various qualitative and quantitative preferences

- Only optimization statements representing objective functions using summation are established components of today's ASP systems
- Example $\#minimize\{40 : sauna, 70 : dive\}$

Approach

- asprin is a framework for handling preferences among the (stable) models of logic programs
  - general because it captures many existing approaches to preference
  - flexible because it allows an easy implementation of new approaches

- asprin builds upon advanced control capacities for multi-shot and meta solving, allowing for

  reducing redundancies

  via ordinary ASP encodings

Approach

- asprin is a framework for handling preferences among the (stable) models of logic programs
  - general because it captures many existing approaches to preference
  - flexible because it allows an easy implementation of new approaches

- asprin builds upon advanced control capacities for multi-shot and meta solving, allowing for
  - reducing redundancies
  - via ordinary ASP encodings

Potassco

Approach

- asprin is a framework for handling preferences among the (stable) models of logic programs
  - general because it captures many existing approaches to preference
  - flexible because it allows an easy implementation of new approaches

- asprin builds upon advanced control capacities for multi-shot and meta solving, allowing for
  - continuous integrated solving process reducing redundancies
  - high customizability via ordinary ASP encodings

Approach

- asprin is a framework for handling preferences among the (stable) models of logic programs
  - general because it captures many existing approaches to preference
  - flexible because it allows an easy implementation of new approaches
- asprin builds upon advanced control capacities for multi-shot and meta solving, allowing for
  - continuous integrated solving process reducing redundancies
  - high customizability via ordinary ASP encodings

Example

Your vacation (logic) program . . .

. . . talking about *sauna*, *dive*, *hike*, *bunji*, *hot*, etc

$\#preference(costs, less(weight))\{40 : sauna, 70 : dive\}$

$\#preference(fun, superset)\{sauna, dive, hike, \neg bunji\}$

$\#preference(temps, aso)\{dive > sauna \parallel hot, sauna > dive \parallel \neg hot\}$

$\#preference(all, pareto)\{name(costs), name(fun), name(temps)\}$

$\#optimize(all)$

Example

Your vacation (logic) program ...

... talking about *sauna*, *dive*, *hike*, *bunji*, *hot*, etc

$\#preference(costs, less(weight))\{40 : sauna, 70 : dive\}$
$\#preference(fun, superset)\{sauna, dive, hike, \neg bunji\}$
$\#preference(temps, aso)\{dive > sauna \parallel hot, sauna > dive \parallel \neg hot\}$
$\#preference(all, pareto)\{name(costs), name(fun), name(temps)\}$
$\#optimize(all)$

Example

Your vacation (logic) program ...

... talking about *sauna*, *dive*, *hike*, *bunji*, *hot*, etc

$\#preference(costs, less(weight))\{40 : sauna, 70 : dive\}$
$\#preference(fun, superset)\{sauna, dive, hike, \neg bunji\}$
$\#preference(temps, aso)\{dive > sauna \parallel hot, sauna > dive \parallel \neg hot\}$
$\#preference(all, pareto)\{name(costs), name(fun), name(temps)\}$
$\#optimize(all)$

Example

Your vacation (logic) program ...

... talking about *sauna*, *dive*, *hike*, *bunji*, *hot*, etc

$\#preference(costs, less(weight))\{40 : sauna, 70 : dive\}$

$\#preference(fun, superset)\{sauna, dive, hike, \neg bunji\}$

$\#preference(temps, aso)\{dive > sauna \parallel hot, sauna > dive \parallel \neg hot\}$

$\#preference(all, pareto)\{name(costs), name(fun), name(temps)\}$

$\#optimize(all)$

Example

Your vacation (logic) program ...

... talking about *sauna*, *dive*, *hike*, *bunji*, *hot*, etc

$\#preference(costs, less(weight))\{40 : sauna, 70 : dive\}$

$\#preference(fun, superset)\{sauna, dive, hike, \neg bunji\}$

$\#preference(temps, aso)\{dive > sauna \parallel hot, sauna > dive \parallel \neg hot\}$

$\#preference(all, pareto)\{name(costs), name(fun), name(temps)\}$

$\#optimize(all)$

Example

Your vacation (logic) program ...

... talking about *sauna*, *dive*, *hike*, *bunji*, *hot*, etc

$\#preference(costs, less(weight))\{40 : sauna, 70 : dive\}$

$\#preference(fun, superset)\{sauna, dive, hike, \neg bunji\}$

$\#preference(temps, aso)\{dive > sauna \parallel hot, sauna > dive \parallel \neg hot\}$

$\#preference(all, pareto)\{name(costs), name(fun), name(temps)\}$

$\#optimize(all)$

Potassco

Example

Your vacation (logic) program ...

... talking about *sauna*, *dive*, *hike*, *bunji*, *hot*, etc

$\#preference(costs, less(weight))\{40 : sauna, 70 : dive\}$

$\#preference(fun, superset)\{sauna, dive, hike, \neg bunji\}$

$\#preference(temps, aso)\{dive > sauna \parallel hot, sauna > dive \parallel \neg hot\}$

$\#preference(all, pareto)\{name(costs), name(fun), name(temps)\}$

$\#optimize(all)$

Outline

Preference

- A strict partial order $\succ$ on the stable models of a logic program
  That is, $X \succ Y$ means that X is preferred to Y
- A stable model X is $\succ$-preferred, if there is no other stable model Y such that $Y \succ X$

- A preference type is a (parametric) class of preference relations
- Example *subset*, *pareto*, etc.

Potassco

Preference

- A strict partial order $\succ$ on the stable models of a logic program
  That is, $X \succ Y$ means that X is preferred to Y
- A stable model X is $\succ$-preferred, if there is no other stable model Y such that $Y \succ X$

- A preference type is a (parametric) class of preference relations
- Example *subset*, *pareto*, etc.

Preference

- A strict partial order $\succ$ on the stable models of a logic program
  That is, $X \succ Y$ means that X is preferred to Y
- A stable model X is $\succ$-preferred, if there is no other stable model Y such that $Y \succ X$

- A preference type is a (parametric) class of preference relations
- Example *subset*, *pareto*, etc.

Potassco

Outline

Language

- naming atom $name(s)$

  where s is the name of a preference
- weighted formula $w_1, \dots, w_l : \phi$

  where each w_i is a term and ϕ is a Boolean formula
- preference element $\Phi_1 > \cdots > \Phi_m \parallel \Phi$

  where each Φ_r is a set of weighted formulas and Φ is a non-weighted formula
- preference statement $\#preference(s,t)\{e_1, \dots, e_n\}$

  where s and t represent the preference statement and its type and each e_j is a preference element
- optimization directive $\#optimize(s)$

  where s is the name of a preference

- preference specification is a set S of preference statements and a directive $\#optimize(s)$ such that S is acyclic, closed, and $s \in S$

Potassco

Language

- naming atom $name(s)$

  where s is the name of a preference
- weighted formula $w_1, \dots, w_l : \phi$

  where each w_i is a term and ϕ is a Boolean formula
- preference element $\Phi_1 > \cdots > \Phi_m \parallel \Phi$

  where each Φ_r is a set of weighted formulas and Φ is a non-weighted formula
- preference statement $\#preference(s, t)\{e_1, \dots, e_n\}$

  where s and t represent the preference statement and its type
  and each e_j is a preference element
- optimization directive $\#optimize(s)$

  where s is the name of a preference

- preference specification is a set S of preference statements and a directive $\#optimize(s)$ such that S is acyclic, closed, and $s \in S$

Language

- naming atom $name(s)$

  where s is the name of a preference
- weighted formula $w_1, \dots, w_l : \phi$

  where each w_i is a term and ϕ is a Boolean formula
- preference element $\Phi_1 > \cdots > \Phi_m \parallel \Phi$

  where each Φ_r is a set of weighted formulas and Φ is a non-weighted formula
- preference statement $\#preference(s, t)\{e_1, \dots, e_n\}$

  where s and t represent the preference statement and its type
  and each e_j is a preference element
- optimization directive $\#optimize(s)$

  where s is the name of a preference

- preference specification is a set S of preference statements and a directive $\#optimize(s)$ such that S is acyclic, closed, and $s \in S$

Language

- naming atom $name(s)$

  where s is the name of a preference
- weighted formula $w_1, \dots, w_l : \phi$

  where each w_i is a term and ϕ is a Boolean formula
- preference element $\Phi_1 > \cdots > \Phi_m \parallel \Phi$

  where each Φ_r is a set of weighted formulas and Φ is a non-weighted formula
- preference statement $\#preference(s, t)\{e_1, \dots, e_n\}$

  where s and t represent the preference statement and its type
  and each e_j is a preference element
- optimization directive $\#optimize(s)$

  where s is the name of a preference

- preference specification is a set S of preference statements and a directive $\#optimize(s)$ such that S is acyclic, closed, and $s \in S$

Language

- naming atom $name(s)$

  where s is the name of a preference
- weighted formula $w_1, \ldots, w_l : \phi$

  where each w_i is a term and ϕ is a Boolean formula
- preference element $\Phi_1 > \cdots > \Phi_m \parallel \Phi$

  where each Φ_r is a set of weighted formulas and Φ is a non-weighted formula
- preference statement $\#preference(s, t)\{e_1, \ldots, e_n\}$

  where s and t represent the preference statement and its type and each e_j is a preference element
- optimization directive $\#optimize(s)$

  where s is the name of a preference

- preference specification is a set S of preference statements and a directive $\#optimize(s)$ such that S is acyclic, closed, and $s \in S$

Potassco

Language

- naming atom $name(s)$

  where s is the name of a preference
- weighted formula $w_1, \dots, w_l : \phi$

  where each w_i is a term and ϕ is a Boolean formula
- preference element $\Phi_1 > \cdots > \Phi_m \parallel \Phi$

  where each Φ_r is a set of weighted formulas and Φ is a non-weighted formula
- preference statement $\#preference(s,t)\{e_1, \dots, e_n\}$

  where s and t represent the preference statement and its type
  and each e_j is a preference element
- optimization directive $\#optimize(s)$

  where s is the name of a preference

- preference specification is a set S of preference statements and a directive $\#optimize(s)$ such that S is acyclic, closed, and $s \in S$

Potassco

Language

- naming atom $name(s)$

  where s is the name of a preference
- weighted formula $w_1, \dots, w_l : \phi$

  where each w_i is a term and ϕ is a Boolean formula
- preference element $\Phi_1 > \cdots > \Phi_m \parallel \Phi$

  where each Φ_r is a set of weighted formulas and Φ is a non-weighted formula
- preference statement $\#preference(s, t)\{e_1, \dots, e_n\}$

  where s and t represent the preference statement and its type
  and each e_j is a preference element
- optimization directive $\#optimize(s)$

  where s is the name of a preference

- preference specification is a set S of preference statements and a directive $\#optimize(s)$ such that S is acyclic, closed, and $s \in S$

Potassco

Preference type

- A preference type t is a function mapping a set of preference elements E to a preference relation

$$t(E) \subseteq \mathcal{A} \times \mathcal{A}$$

- Examples

$$(X, Y) \in \mathit{subset}(E) \quad \text{iff} \quad \{\ell \in E \mid X \models \ell\} \subset \{\ell \in E \mid Y \models \ell\}$$

$$(X, Y) \in \mathit{pareto}(E) \quad \text{iff} \quad \bigwedge_{name(s) \in E}(X \succeq_s Y) \wedge \bigvee_{name(s) \in E}(X \succ_s Y)$$

Potassco

Preference type

- A preference type t is a function mapping a set of preference elements E to a preference relation

$$t(E) \subseteq \mathcal{A} \times \mathcal{A}$$

- Examples
  - $(X, Y) \in \mathit{subset}(E)$ iff $\{\ell \in E \mid X \models \ell\} \subset \{\ell \in E \mid Y \models \ell\}$
  - $(X, Y) \in \mathit{pareto}(E)$ iff $\bigwedge_{name(s) \in E}(X \succeq_s Y) \wedge \bigvee_{name(s) \in E}(X \succ_s Y)$

Potassco

Preference relation

- A preference relation is obtained by applying a preference type to a set of preference elements

  $\#preference(s,t)\,E$ declares preference relation $t(E)$, denoted by $\succ_s$

- Example $\#preference(1, subset)\{a,b,c\}$ declares

  $$X \succ_1 Y \text{ iff } \{\ell \in \{a,b,c\} \mid X \models \ell\} \subset \{\ell \in \{a,b,c\} \mid Y \models \ell\}$$

Potassco

Preference relation

- A preference relation is obtained by applying a preference type to a set of preference elements

  $\#preference(s, t)\, E$ declares preference relation $t(E)$, denoted by $\succ_s$

- Example $\#preference(1, subset)\{a, b, c\}$ declares

  $$X \succ_1 Y \text{ iff } \{\ell \in \{a, b, c\} \mid X \models \ell\} \subset \{\ell \in \{a, b, c\} \mid Y \models \ell\}$$

Preference relation

- A preference relation is obtained by applying a preference type to a set of preference elements

  $\#preference(s, t)\, E$ declares preference relation $t(E)$, denoted by $\succ_s$

- Example $\#preference(1, subset)\{a, b, c\}$ declares

  $$X \succ_1 Y \text{ iff } \{\ell \in \{a, b, c\} \mid X \models \ell\} \subset \{\ell \in \{a, b, c\} \mid Y \models \ell\}$$

Outline

Preference program

- Reification $H_X = \{holds(a) \mid a \in X\}$ and $H'_X = \{holds'(a) \mid a \in X\}$
- Preference program Let s be a preference statement declaring $\succ_s$.

  We define Q_s as a preference program for s, if for all sets $X, Y \subseteq \mathcal{A}$, we have

  $$X \succ_s Y \quad \text{iff} \quad Q_s \cup H_X \cup H'_Y \text{ is satisfiable}$$

- Note Q_s is implemented as $F_s \cup E_{t_s} \cup C$
- Note *asprin*'s expressiveness is delineated by the decision problem

Potassco

Preference program

- Reification $H_X = \{holds(a) \mid a \in X\}$ and $H'_X = \{holds'(a) \mid a \in X\}$
- Preference program Let s be a preference statement declaring $\succ_s$.

  We define Q_s as a preference program for s, if for all sets $X, Y \subseteq \mathcal{A}$, we have

$$X \succ_s Y \ \text{ iff } \ Q_s \cup H_X \cup H'_Y \text{ is satisfiable}$$

- Note Q_s is implemented as $F_s \cup E_{t_s} \cup C$
- Note *asprin*'s expressiveness is delineated by the decision problem

Potassco

Preference program

- Reification $H_X = \{holds(a) \mid a \in X\}$ and $H'_X = \{holds'(a) \mid a \in X\}$
- Preference program Let s be a preference statement declaring $\succ_s$.

  We define Q_s as a preference program for s, if for all sets $X, Y \subseteq \mathcal{A}$, we have

$$X \succ_s Y \text{ iff } Q_s \cup H_X \cup H'_Y \text{ is satisfiable}$$

- Note Q_s is implemented as $F_s \cup E_{t_s} \cup C$
- Note *asprin*'s expressiveness is delineated by the decision problem

Potassco

Preference program

- Reification $H_X = \{holds(a) \mid a \in X\}$ and $H'_X = \{holds'(a) \mid a \in X\}$
- Preference program Let s be a preference statement declaring $\succ_s$.

  We define Q_s as a preference program for s, if for all sets $X, Y \subseteq \mathcal{A}$, we have

  $$X \succ_s Y \text{ iff } Q_s \cup H_X \cup H'_Y \text{ is satisfiable}$$

- Note Q_s is implemented as $F_s \cup E_{t_s} \cup C$
- Note *asprin*'s expressiveness is delineated by the decision problem

$$\#preference(1, subset)\{a, b, c\}$$
$$\#optimize(1)$$

$H_{\{a\}} = \{$
```
holds(a).
```
$\}$

$H'_{\{a,b\}} = \{$
```
holds'(a).  holds'(b).
```
$\}$

$F_1 = \{$
```
preference(1,subset).        preference(1,1,1,for(a),()).
                             preference(1,2,1,for(b),()).
optimize(1).                 preference(1,3,1,for(c),()).
```
$\}$

$E_{subset} = \{$
```
better(P) :- preference(P,subset),
  not holds(X),      holds'(X), preference(P,_,_,for(X),_),
  not holds(Y) : not holds'(Y), preference(P,_,_,for(Y),_).
```
$\}$

$C = \{$
```
:- optimize(P), not better(P).
```
$\}$

There is a stable model, indicating that $\{a\} \succ_1 \{a, b\}$

Potassco

$$\#preference(1, subset)\{a, b, c\}$$
$$\#optimize(1)$$

$H_{\{a\}} = \{$
```
holds(a).
```
$\}$

$H'_{\{a,b\}} = \{$
```
holds'(a).  holds'(b).
```
$\}$

$F_1 = \{$
```
preference(1,subset).        preference(1,1,1,for(a),()).
                             preference(1,2,1,for(b),()).
optimize(1).                 preference(1,3,1,for(c),()).
```
$\}$

$E_{subset} = \{$
```
better(P) :- preference(P,subset),
  not holds(X),      holds'(X), preference(P,_,_,for(X),_),
  not holds(Y) : not holds'(Y), preference(P,_,_,for(Y),_).
```
$\}$

$C = \{$
```
:- optimize(P), not better(P).
```
$\}$

There is a stable model, indicating that $\{a\} \succ_1 \{a, b\}$

Potassco

$$\#preference(1, subset)\{a, b, c\}$$
$$\#optimize(1)$$

$H_{\{a\}}$ = {
```
holds(a).
```
}

$H'_{\{a,b\}}$ = {
```
holds'(a).  holds'(b).
```
}

F_1 = {
```
preference(1,subset).      preference(1,1,1,for(a),()).
                           preference(1,2,1,for(b),()).
optimize(1).               preference(1,3,1,for(c),()).
```
}

E_{subset} = {
```
better(P) :- preference(P,subset),
  not holds(X),      holds'(X), preference(P,_,_,for(X),_),
  not holds(Y) : not holds'(Y), preference(P,_,_,for(Y),_).
```
}

C = {
```
:- optimize(P), not better(P).
```
}

There is a stable model, indicating that $\{a\} \succ_1 \{a, b\}$

$$\#preference(1, subset)\{a, b, c\}$$
$$\#optimize(1)$$

$H_{\{a,b\}} = \{$
```
holds(a). holds(b).
```
$\}$

$H'_{\{a\}} = \{$
```
holds'(a).
```
$\}$

$F_1 = \{$
```
preference(1,subset).       preference(1,1,1,for(a),()).
                            preference(1,2,1,for(b),()).
optimize(1).                preference(1,3,1,for(c),()).
```
$\}$

$E_{subset} = \{$
```
better(P) :- preference(P,subset),
  not holds(X),      holds'(X), preference(P,_,_,for(X),_),
  not holds(Y) : not holds'(Y), preference(P,_,_,for(Y),_).
```
$\}$

$C = \{$
```
:- optimize(P), not better(P).
```
$\}$

There is no stable model, indicating that $\{a, b\} \not\succ_1 \{a\}$

Potassco

Basic algorithm $\mathit{solveOpt}(P, s)$

Input : A program P over $\mathcal{A}$ and preference statement s
Output : A $\succ_s$-preferred stable model of P, if P is satisfiable, and $\bot$ otherwise

$Y \leftarrow \mathit{solve}(P)$
if $Y = \bot$ **then return** $\bot$
repeat
  $X \leftarrow Y$
  $Y \leftarrow \mathit{solve}\big(P \cup Q_s \cup R \cup H'_X\big)$
until $Y = \bot$
return X

where $R = \{\mathit{holds}(a) \leftarrow a \mid a \in \mathcal{A}\}$

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)
- Composite preference types
  - `neg`
  - `and`
  - `pareto`
  - `lexico`
- Customized preference types

☞ *See paper (and Potassco Guide) on how to define new types!*

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)
- Composite preference types
  - `neg`
  - `and`
  - `pareto`
  - `lexico`
- Customized preference types

☞ *See paper (and Potassco Guide) on how to define new types!*

Potassco

Outline

Potassco

Available in *asprin*, continued

- Weak constraints (DLV) and minimize statements (SMODELS)
- Answer set optimization (Brewka, Niemelä & Truszczyński; IJCAI)
  - Answer set optimization with penalties (Brewka; KR)
  - Ordered disjunctions (Brewka; AAAI)
- Qualitative preferences (Di Rosa, Giunchiglia & Maratea; Constraints)
- Preference language (Son & Pontelli; TPLP)

☞ *See paper for details and experiments next!*

Available in *asprin*, continued

- Weak constraints (DLV) and minimize statements (SMODELS)
- Answer set optimization (Brewka, Niemelä & Truszczyński; IJCAI)
  - Answer set optimization with penalties (Brewka; KR)
  - Ordered disjunctions (Brewka; AAAI)
- Qualitative preferences (Di Rosa, Giunchiglia & Maratea; Constraints)
- Preference language (Son & Pontelli; TPLP)

☞ *See paper for details and experiments next!*

Outline

Potassco

Outline

Heuristic framework

- *clingo* allows for incorporating domain-specific heuristics into ASP solving
  - input language for expressing domain-specific heuristics
  - solving capacities for integrating domain-specific heuristics
- Heuristic predicate `_heuristic`
- Heuristic modifiers (atom, *a*, and integer, *v*)
  - `init` for initializing the heuristic value of *a* with *v*
  - `factor` for amplifying the heuristic value of *a* by factor *v*
  - `level` for ranking all atoms; the rank of *a* is *v*
  - `sign` for attributing the sign of *v* as truth value to *a*
- Heuristic atoms

```
_heuristic(occurs(A,T),factor,T) :- action(A), time(T).
```

Heuristic framework

- *clingo* allows for incorporating domain-specific heuristics into ASP solving
  - input language for expressing domain-specific heuristics
  - solving capacities for integrating domain-specific heuristics
- Heuristic predicate `_heuristic`
- Heuristic modifiers (atom, *a*, and integer, *v*)
  - `init` for initializing the heuristic value of *a* with *v*
  - `factor` for amplifying the heuristic value of *a* by factor *v*
  - `level` for ranking all atoms; the rank of *a* is *v*
  - `sign` for attributing the sign of *v* as truth value to *a*
- Heuristic atoms

```
_heuristic(occurs(A,T),factor,T) :- action(A), time(T).
```

Potassco

Heuristic framework

- *clingo* allows for incorporating domain-specific heuristics into ASP solving
  - input language for expressing domain-specific heuristics
  - solving capacities for integrating domain-specific heuristics
- Heuristic predicate `_heuristic`
- Heuristic modifiers (atom, *a*, and integer, *v*)

  `init` for initializing the heuristic value of *a* with *v*

  `factor` for amplifying the heuristic value of *a* by factor *v*

  `level` for ranking all atoms; the rank of *a* is *v*

  `sign` for attributing the sign of *v* as truth value to *a*

- Heuristic atoms

```
_heuristic(occurs(A,T),factor,T) :- action(A), time(T).
```

Heuristic framework

- *clingo* allows for incorporating domain-specific heuristics into ASP solving
  - input language for expressing domain-specific heuristics
  - solving capacities for integrating domain-specific heuristics
- Heuristic predicate `_heuristic`
- Heuristic modifiers (atom, *a*, and integer, *v*)

`init`	for initializing the heuristic value of *a* with *v*
`factor`	for amplifying the heuristic value of *a* by factor *v*
`level`	for ranking all atoms; the rank of *a* is *v*
`sign`	for attributing the sign of *v* as truth value to *a*

- Heuristic atoms

```
_heuristic(occurs(A,T),factor,T) :- action(A), time(T).
```

Potassco

Heuristic framework

- *clingo* allows for incorporating domain-specific heuristics into ASP solving
  - input language for expressing domain-specific heuristics
  - solving capacities for integrating domain-specific heuristics
- Heuristic predicate `_heuristic`
- Heuristic modifiers (atom, *a*, and integer, *v*)

  `init` for initializing the heuristic value of *a* with *v*
  `factor` for amplifying the heuristic value of *a* by factor *v*
  `level` for ranking all atoms; the rank of *a* is *v*
  `sign` for attributing the sign of *v* as truth value to *a*

- Heuristic atoms

```
_heuristic(occurs(mv,5),factor,5) :- action(mv), time(5).
```

asprin with different heuristic settings

Benchmark \ System			*asprin*$_w$	*asprin*$_w$+s	*asprin*$_w$+l	*asprin*$_w$+f	*asprin*$_s$	*asprin*$_s$+s	*asprin*$_s$+l	*asprin*$_s$+f
Ricochet	(30)	20.00	432 (8, 4)	407 (7, 4)	68 (1, 0)	71 (1, 0)	365 (8, 3)	461 (7, 10)	69 (1, 0)	71 (1, 0)
Timetabling	(12)	23687.75	345 (285, 3)	255 (202, 2)	900 (4, 12)	6 (1, 0)	217 (144, 2)	21 (18, 0)	900 (2, 12)	5 (1, 0)
Puzzle	(7)	580.57	82 (2, 0)	112 (2, 0)	136 (2, 0)	416 (2, 1)	31 (1, 0)	32 (1, 0)	21 (1, 0)	51 (1, 0)
Crossing	(24)	211.92	104 (42, 1)	98 (35, 0)	805 (19, 20)	387 (6, 6)	0 (6, 0)	1 (6, 0)	7 (9, 0)	3 (1, 0)
Valves	(30)	56.63	69 (7, 0)	65 (6, 0)	460 (8, 11)	715 (0, 22)	38 (4, 0)	39 (4, 0)	339 (4, 6)	673 (0, 21)
Expansion	(30)	7501.87	216 (299, 0)	10 (15, 0)	38 (7, 0)	12 (3, 0)	64 (295, 0)	14 (54, 0)	4 (4, 0)	3 (1, 0)
Repair	(30)	6750.73	76 (48, 0)	15 (47, 0)	71 (3, 2)	8 (2, 0)	8 (43, 0)	3 (11, 0)	1 (1, 0)	1 (1, 0)
Diagnosis	(30)	1669.00	196 (341, 3)	76 (66, 0)	43 (4, 0)	118 (3, 2)	19 (338, 0)	2 (39, 0)	0 (1, 0)	0 (1, 0)
$\varnothing(\varnothing, \Sigma)$			190 (129, 11)	130 (48, 6)	315 (6, 45)	217 (2, 31)	93 (105, 5)	72 (18, 10)	168 (3, 18)	101 (1, 21)

- *asprin*$_w$
- *asprin*$_w$+s
- *asprin*$_w$+l
- *asprin*$_w$+f

- w — weight-based
- s — sign heuristic
- l — level-based heuristic
- f — factor-based heuristic

- *asprin*$_s$
- *asprin*$_s$+s
- *asprin*$_s$+l
- *asprin*$_s$+f

- s — subset-based
- s — sign heuristic
- l — level-based heuristic
- f — factor-based heuristic

asprin with different heuristic settings

Benchmark \ System			*asprin*$_w$	*asprin*$_w$+s	*asprin*$_w$+l	*asprin*$_w$+f	*asprin*$_s$	*asprin*$_s$+s	*asprin*$_s$+l	*asprin*$_s$+f
Ricochet	(30)	20.00	432 (8, 4)	407 (7, 4)	68 (1, 0)	71 (1, 0)	365 (8, 3)	461 (7, 10)	69 (1, 0)	71 (1, 0)
Timetabling	(12)	23687.75	345 (285, 3)	255 (202, 2)	900 (4, 12)	6 (1, 0)	217 (144, 2)	21 (18, 0)	900 (2, 12)	5 (1, 0)
Puzzle	(7)	580.57	82 (2, 0)	112 (2, 0)	136 (2, 0)	416 (2, 1)	31 (1, 0)	32 (1, 0)	21 (1, 0)	51 (1, 0)
Crossing	(24)	211.92	104 (42, 1)	98 (35, 0)	805 (19, 20)	387 (6, 6)	0 (6, 0)	1 (6, 0)	7 (9, 0)	3 (1, 0)
Valves	(30)	56.63	69 (7, 0)	65 (6, 0)	460 (8, 11)	715 (0, 22)	38 (4, 0)	39 (4, 0)	339 (4, 6)	673 (0, 21)
Expansion	(30)	7501.87	216 (299, 0)	10 (15, 0)	38 (7, 0)	12 (3, 0)	64 (295, 0)	14 (54, 0)	4 (4, 0)	3 (1, 0)
Repair	(30)	6750.73	76 (48, 0)	15 (47, 0)	71 (3, 2)	8 (2, 0)	8 (43, 0)	3 (11, 0)	1 (1, 0)	1 (1, 0)
Diagnosis	(30)	1669.00	196 (341, 3)	76 (66, 0)	43 (4, 0)	118 (3, 2)	19 (338, 0)	2 (39, 0)	0 (1, 0)	0 (1, 0)
$\varnothing(\varnothing, \Sigma)$			190 (129, 11)	130 (48, 6)	315 (6, 45)	217 (2, 31)	93 (105, 5)	72 (18, 10)	168 (3, 18)	101 (1, 21)

- *asprin*$_w$
- *asprin*$_w$+s
- *asprin*$_w$+l
- *asprin*$_w$+f

- w — weight-based
- s — sign heuristic
- l — level-based heuristic
- f — factor-based heuristic

- *asprin*$_s$
- *asprin*$_s$+s
- *asprin*$_s$+l
- *asprin*$_s$+f

- s — subset-based
- s — sign heuristic
- l — level-based heuristic
- f — factor-based heuristic

asprin with different heuristic settings

Benchmark \ System			*asprin*$_w$	*asprin*$_w$+s	*asprin*$_w$+l	*asprin*$_w$+f	*asprin*$_s$	*asprin*$_s$+s	*asprin*$_s$+l	*asprin*$_s$+f
Ricochet	(30)	20.00	432 (8, 4)	407 (7, 4)	68 (1, 0)	71 (1, 0)	365 (8, 3)	461 (7, 10)	69 (1, 0)	71 (1, 0)
Timetabling	(12)	23687.75	345 (285, 3)	255 (202, 2)	900 (4, 12)	6 (1, 0)	217 (144, 2)	21 (18, 0)	900 (2, 12)	5 (1, 0)
Puzzle	(7)	580.57	82 (2, 0)	112 (2, 0)	136 (2, 0)	416 (2, 1)	31 (1, 0)	32 (1, 0)	21 (1, 0)	51 (1, 0)
Crossing	(24)	211.92	104 (42, 1)	98 (35, 0)	805 (19, 20)	387 (6, 6)	0 (6, 0)	1 (6, 0)	7 (9, 0)	3 (1, 0)
Valves	(30)	56.63	69 (7, 0)	65 (6, 0)	460 (8, 11)	715 (0, 22)	38 (4, 0)	39 (4, 0)	339 (4, 6)	673 (0, 21)
Expansion	(30)	7501.87	216 (299, 0)	10 (15, 0)	38 (7, 0)	12 (3, 0)	64 (295, 0)	14 (54, 0)	4 (4, 0)	3 (1, 0)
Repair	(30)	6750.73	76 (48, 0)	15 (47, 0)	71 (3, 2)	8 (2, 0)	8 (43, 0)	3 (11, 0)	1 (1, 0)	1 (1, 0)
Diagnosis	(30)	1669.00	196 (341, 3)	76 (66, 0)	43 (4, 0)	118 (3, 2)	19 (338, 0)	2 (39, 0)	0 (1, 0)	0 (1, 0)
$\varnothing(\varnothing, \Sigma)$			190 (129, 11)	130 (48, 6)	315 (6, 45)	217 (2, 31)	93 (105, 5)	72 (18, 10)	168 (3, 18)	101 (1, 21)

- *asprin*$_w$
- *asprin*$_w$+s
- *asprin*$_w$+l
- *asprin*$_w$+f

- w — weight-based
- s — sign heuristic
- l — level-based heuristic
- f — factor-based heuristic

- *asprin*$_s$
- *asprin*$_s$+s
- *asprin*$_s$+l
- *asprin*$_s$+f

- s — subset-based
- s — sign heuristic
- l — level-based heuristic
- f — factor-based heuristic

Outline

Potassco

asprin versus *clingo* and *metasp*

(B,D,R&S; AAAI)

Benchmark \ System			*clingo*	$asprin_w$	$asprin_w$ -f	*metasp*	$asprin_s$	$asprin_s$ -f
Ricochet	(30)	20.00	104.74 (0)	174.26 (0)	113.45 (0)	811.32 (24)	175.71 (0)	109.91 (0)
Timetabling	(12)	23687.75	35.82 (0)	490.39 (5)	694.92 (8)	798.75 (10)	142.03 (0)	12.01 (0)
Puzzle	(7)	580.57	77.00 (0)	77.39 (0)	96.70 (0)	34.79 (0)	17.06 (0)	17.22 (0)
Crossing	(24)	211.92	48.43 (0)	105.64 (1)	67.50 (0)	62.33 (0)	0.50 (0)	0.46 (0)
Valves	(30)	56.63	52.53 (0)	72.85 (0)	78.11 (0)	900.00 (30)	45.01 (0)	39.31 (0)
Expansion	(30)	7501.87	91.53 (0)	373.56 (2)	241.05 (7)	900.00 (30)	292.57 (0)	21.12 (0)
Repair	(30)	6750.73	71.78 (0)	102.19 (0)	43.94 (0)	900.00 (30)	6.88 (0)	2.19 (0)
Diagnosis	(30)	1669.00	84.96 (0)	254.19 (3)	101.33 (0)	181.71 (6)	41.55 (0)	1.56 (0)
$\varnothing(\Sigma)$			70.85 (0)	206.31 (11)	179.63 (15)	573.61 (130)	90.16 (0)	25.47 (0)

- *clingo* (using branch-and-bound)
- $asprin_w$
- $asprin_w$ -f

- w — weight-based
- -f — no phase saving

- *metasp* (using disjunction)
- $asprin_s$
- $asprin_s$ -f

- s — subset-based
- -f — no phase saving

asprin versus *clingo* and *metasp*

(B,D,R&S; AAAI)

Benchmark \ System			*clingo*	$asprin_w$	$asprin_w$ -f	*metasp*	$asprin_s$	$asprin_s$ -f
Ricochet	(30)	20.00	104.74 (0)	174.26 (0)	113.45 (0)	811.32 (24)	175.71 (0)	109.91 (0)
Timetabling	(12)	23687.75	35.82 (0)	490.39 (5)	694.92 (8)	798.75 (10)	142.03 (0)	12.01 (0)
Puzzle	(7)	580.57	77.00 (0)	77.39 (0)	96.70 (0)	34.79 (0)	17.06 (0)	17.22 (0)
Crossing	(24)	211.92	48.43 (0)	105.64 (1)	67.50 (0)	62.33 (0)	0.50 (0)	0.46 (0)
Valves	(30)	56.63	52.53 (0)	72.85 (0)	78.11 (0)	900.00 (30)	45.01 (0)	39.31 (0)
Expansion	(30)	7501.87	91.53 (0)	373.56 (2)	241.05 (7)	900.00 (30)	292.57 (0)	21.12 (0)
Repair	(30)	6750.73	71.78 (0)	102.19 (0)	43.94 (0)	900.00 (30)	6.88 (0)	2.19 (0)
Diagnosis	(30)	1669.00	84.96 (0)	254.19 (3)	101.33 (0)	181.71 (6)	41.55 (0)	1.56 (0)
$\varnothing(\Sigma)$			70.35 (0)	206.31 (11)	179.63 (15)	573.61 (130)	90.16 (0)	25.47 (0)

- ✓ *clingo* (using branch-and-bound)
- $asprin_w$
- $asprin_w$ -f
- w — weight-based
- -f — no phase saving

- *metasp* (using disjunction)
- $asprin_s$
- ✓ $asprin_s$ -f
- s — subset-based
- -f — no phase saving

aso versus *asprin*

n	*aso*	aso_l	$asprin_a$	$asprin_{l+a}$
350	9 (0)	17 (0)	4 (0)	5 (0)
360	14 (0)	22 (0)	48 (0)	50 (0)
370	15 (0)	25 (0)	38 (0)	39 (0)
380	10 (0)	23 (0)	8 (0)	9 (0)
390	59 (0)	72 (0)	50 (1)	52 (1)
400	22 (0)	33 (0)	28 (0)	30 (0)
410	87 (1)	96 (1)	124 (2)	125 (2)
420	97 (1)	108 (1)	60 (0)	62 (0)
430	68 (0)	79 (0)	144 (0)	147 (0)
440	165 (3)	175 (3)	165 (2)	167 (2)
450	45 (0)	61 (0)	52 (0)	54 (0)
460	112 (1)	125 (1)	117 (2)	120 (2)
470	201 (4)	210 (4)	161 (2)	162 (2)
480	152 (2)	165 (2)	70 (1)	72 (1)
490	206 (2)	218 (2)	265 (4)	267 (4)
$\varnothing(\Sigma)$	84 (14)	95 (14)	89 (14)	91 (14)

- *aso* — dedicated system
- aso_l — dedicated system with level-based heursitic
- $asprin_a$
- $asprin_{l+a}$ — with level-based heursitic

Potassco

satpref versus *asprin*

Benchmark\System	*satpref*	*satpref*+s	*satpref*+H	*asprin*$_p$	*asprin*$_p$+s	*asprin*$_p$+H
0.0	0 (29, 0)	0 (1, 0)	0 (1, 0)	1 (16, 0)	0 (2, 0)	0 (1, 0)
0.00621	0 (35, 0)	0 (1, 0)	90 (1, 6)	1 (17, 0)	1 (2, 0)	1 (1, 0)
0.01243	1 (75, 0)	1 (3, 0)	118 (1, 7)	6 (26, 0)	2 (3, 0)	3 (1, 0)
0.02486	8 (388, 0)	6 (10, 0)	635 (1, 38)	55 (74, 0)	9 (8, 0)	64 (1, 4)
0.04972	67 (1463, 2)	16 (36, 0)	900 (0,100)	318 (203, 16)	26 (17, 0)	176 (1,14)
1.0	850 (10315,88)	243 (590,10)	177 (1, 12)	856 (323, 92)	174 (96, 0)	280 (1,24)
$\varnothing(\varnothing, \Sigma)$	154 (2051,90)	44 (107,10)	320 (1,163)	206 (110,108)	35 (21, 0)	88 (1,42)
MAXSAT	54 (8849, 0)	9 (7, 0)	62 (1, 0)	835 (957, 31)	109 (31, 3)	171 (1, 6)
PBO/pbo-mqc-nencdr	5 (267, 0)	2 (2, 0)	664 (1, 88)	150 (207, 14)	9 (2, 0)	244 (1,20)
PBO/pbo-mqc-nlogencdr	3 (228, 0)	1 (2, 0)	237 (1, 21)	110 (214, 3)	5 (2, 0)	141 (1,15)
PSEUDO/primes	110 (396,18)	110 (1,18)	110 (1, 18)	215 (334, 27)	106 (5,17)	110 (1,17)
PSEUDO/routing	346 (409, 4)	49 (1, 0)	50 (1, 0)	85 (475, 0)	4 (1, 0)	86 (1, 1)
Partial-MINONE	14 (2, 0)	14 (2, 0)	7 (1, 0)	24 (2, 0)	24 (1, 0)	25 (1, 0)
$\varnothing(\varnothing, \Sigma)$	88 (1692,22)	31 (2,18)	188 (1,127)	236 (365, 75)	43 (7,20)	129 (1,59)

- *satpref*
- *satpref*+s
- *satpref*+H

- s — sign heuristic
- H — complex heuristic

- *asprin*$_p$
- *asprin*$_p$+s
- *asprin*$_p$+H

- s — sign heuristic
- H — complex heuristic

Potassco

Outline

Potassco

Summary

- asprin stands for "ASP for Preference handling"
- asprin is a general, flexible, and extendable framework for preference handling in ASP
- asprin caters to
  - off-the-shelf users using the preference relations in *asprin*'s library
  - preference engineers customizing their own preference relations
- asprin can be boosted by *clingo*'s heuristic framework
- `http://potassco.sourceforge.net/labs.html#asprin`

Summary

- asprin stands for “ASP for Preference handling”
- asprin is a general, flexible, and extendable framework for preference handling in ASP
- asprin caters to
  - off-the-shelf users using the preference relations in *asprin*’s library
  - preference engineers customizing their own preference relations
- asprin can be boosted by *clingo*’s heuristic framework
- `http://potassco.sourceforge.net/labs.html#asprin`

Potassco

Summary

- asprin stands for "ASP for Preference handling"
- asprin is a general, flexible, and extendable framework for preference handling in ASP
- asprin caters to
  - off-the-shelf users using the preference relations in *asprin*'s library
  - preference engineers customizing their own preference relations
- asprin can be boosted by *clingo*'s heuristic framework
- http://potassco.sourceforge.net/labs.html#asprin

Potassco

Summary

- asprin stands for "ASP for Preference handling"
- asprin is a general, flexible, and extendable framework for preference handling in ASP
- asprin caters to
  - off-the-shelf users using the preference relations in *asprin*'s library
  - preference engineers customizing their own preference relations
- asprin can be boosted by *clingo*'s heuristic framework

- http://potassco.sourceforge.net/labs.html#asprin

Potassco

Summary

- asprin stands for "ASP for Preference handling"
- asprin is a general, flexible, and extendable framework for preference handling in ASP
- asprin caters to
  - off-the-shelf users using the preference relations in *asprin*'s library
  - preference engineers customizing their own preference relations
- asprin can be boosted by *clingo*'s heuristic framework
- `http://potassco.sourceforge.net/labs.html#asprin`

Potassco