

Answer Set Solving in Practice

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



Potassco Slide Packages are licensed under a Creative Commons Attribution 3.0 Unported License.

Language: Overview

- 1 Motivation
- 2 Core language
- 3 Extended language
- 4 Intermediate formats

Outline

- 1 Motivation
- 2 Core language
- 3 Extended language
- 4 Intermediate formats

Basic language extensions

- The expressiveness of a language can be enhanced by introducing new constructs
- To this end, we must address the following issues:
 - What is the **syntax** of the new language construct?
 - What is the **semantics** of the new language construct?
 - How to **implement** the new language construct?
- A way of providing semantics is to furnish a translation removing the new constructs, eg. classical negation
- This translation might also be used for implementing the language extension

Basic language extensions

- The expressiveness of a language can be enhanced by introducing new constructs
- To this end, we must address the following issues:
 - What is the **syntax** of the new language construct?
 - What is the **semantics** of the new language construct?
 - How to **implement** the new language construct?
- A way of providing semantics is to furnish a translation removing the new constructs, eg. classical negation
- This translation might also be used for implementing the language extension

Basic language extensions

- The expressiveness of a language can be enhanced by introducing new constructs
- To this end, we must address the following issues:
 - What is the **syntax** of the new language construct?
 - What is the **semantics** of the new language construct?
 - How to **implement** the new language construct?
- A way of providing semantics is to furnish a translation removing the new constructs, eg. classical negation
- This translation might also be used for implementing the language extension

Outline

- 1 Motivation
- 2 Core language
- 3 Extended language
- 4 Intermediate formats

Outline

- 1 Motivation
- 2 Core language
 - Integrity constraint
 - Choice rule
 - Cardinality rule
 - Weight rule
- 3 Extended language
- 4 Intermediate formats

Integrity constraint

- Idea Eliminate unwanted solution candidates
- Syntax An **integrity constraint** is of the form

$$\leftarrow a_1, \dots, a_m, \sim a_{m+1}, \dots, \sim a_n$$

where $0 \leq m \leq n$ and each a_i is an atom for $1 \leq i \leq n$

- Example `:- edge(3,7), color(3,red), color(7,red).`
- Embedding The above integrity constraint can be turned into the normal rule

$$x \leftarrow a_1, \dots, a_m, \sim a_{m+1}, \dots, \sim a_n, \sim x$$

where x is a new symbol, that is, $x \notin \mathcal{A}$.

- Another example $P = \{a \leftarrow \sim b, b \leftarrow \sim a\}$
versus $P' = P \cup \{\leftarrow a\}$ and $P'' = P \cup \{\leftarrow \sim a\}$

Integrity constraint

- Idea Eliminate unwanted solution candidates
- Syntax An **integrity constraint** is of the form

$$\leftarrow a_1, \dots, a_m, \sim a_{m+1}, \dots, \sim a_n$$

where $0 \leq m \leq n$ and each a_i is an atom for $1 \leq i \leq n$

- Example `:- edge(3,7), color(3,red), color(7,red).`
- Embedding The above integrity constraint can be turned into the normal rule

$$x \leftarrow a_1, \dots, a_m, \sim a_{m+1}, \dots, \sim a_n, \sim x$$

where x is a new symbol, that is, $x \notin \mathcal{A}$.

- Another example $P = \{a \leftarrow \sim b, b \leftarrow \sim a\}$
versus $P' = P \cup \{\leftarrow a\}$ and $P'' = P \cup \{\leftarrow \sim a\}$

Integrity constraint

- Idea Eliminate unwanted solution candidates
- Syntax An **integrity constraint** is of the form

$$\leftarrow a_1, \dots, a_m, \sim a_{m+1}, \dots, \sim a_n$$

where $0 \leq m \leq n$ and each a_i is an atom for $1 \leq i \leq n$

- Example `:- edge(3,7), color(3,red), color(7,red).`
- Embedding The above integrity constraint can be turned into the normal rule

$$x \leftarrow a_1, \dots, a_m, \sim a_{m+1}, \dots, \sim a_n, \sim x$$

where x is a new symbol, that is, $x \notin \mathcal{A}$.

- Another example $P = \{a \leftarrow \sim b, b \leftarrow \sim a\}$
versus $P' = P \cup \{\leftarrow a\}$ and $P'' = P \cup \{\leftarrow \sim a\}$

Outline

- 1 Motivation
- 2 Core language
 - Integrity constraint
 - Choice rule
 - Cardinality rule
 - Weight rule
- 3 Extended language
- 4 Intermediate formats

Choice rule

- Idea Choices over subsets
- Syntax A **choice rule** is of the form

$$\{a_1, \dots, a_m\} \leftarrow a_{m+1}, \dots, a_n, \sim a_{n+1}, \dots, \sim a_o$$

where $0 \leq m \leq n \leq o$ and each a_i is an atom for $1 \leq i \leq o$

- Informal meaning If the body is satisfied by the stable model at hand, then any subset of $\{a_1, \dots, a_m\}$ can be included in the stable model
- Example $\{ \text{buy}(\text{pizza}); \text{buy}(\text{wine}); \text{buy}(\text{corn}) \} \text{ :- at}(\text{grocery})$.
- Another example $P = \{ \{a\} \leftarrow b, b \leftarrow \}$ has two stable models: $\{b\}$ and $\{a, b\}$

Choice rule

- Idea Choices over subsets
- Syntax A **choice rule** is of the form

$$\{a_1, \dots, a_m\} \leftarrow a_{m+1}, \dots, a_n, \sim a_{n+1}, \dots, \sim a_o$$

where $0 \leq m \leq n \leq o$ and each a_i is an atom for $1 \leq i \leq o$

- Informal meaning If the body is satisfied by the stable model at hand, then any subset of $\{a_1, \dots, a_m\}$ can be included in the stable model
- Example $\{ \text{buy}(\text{pizza}); \text{buy}(\text{wine}); \text{buy}(\text{corn}) \} \text{ :- } \text{at}(\text{grocery})$.
- Another example $P = \{ \{a\} \leftarrow b, b \leftarrow \}$ has two stable models: $\{b\}$ and $\{a, b\}$

Choice rule

- Idea Choices over subsets
- Syntax A **choice rule** is of the form

$$\{a_1, \dots, a_m\} \leftarrow a_{m+1}, \dots, a_n, \sim a_{n+1}, \dots, \sim a_o$$

where $0 \leq m \leq n \leq o$ and each a_i is an atom for $1 \leq i \leq o$

- Informal meaning If the body is satisfied by the stable model at hand, then any subset of $\{a_1, \dots, a_m\}$ can be included in the stable model
- Example $\{ \text{buy}(\text{pizza}); \text{buy}(\text{wine}); \text{buy}(\text{corn}) \} \text{ :- at}(\text{grocery})$.
- Another example $P = \{ \{a\} \leftarrow b, b \leftarrow \}$ has two stable models: $\{b\}$ and $\{a, b\}$

Choice rule

- Idea Choices over subsets
- Syntax A **choice rule** is of the form

$$\{a_1, \dots, a_m\} \leftarrow a_{m+1}, \dots, a_n, \sim a_{n+1}, \dots, \sim a_o$$

where $0 \leq m \leq n \leq o$ and each a_i is an atom for $1 \leq i \leq o$

- Informal meaning If the body is satisfied by the stable model at hand, then any subset of $\{a_1, \dots, a_m\}$ can be included in the stable model
- Example $\{ \text{buy}(\text{pizza}); \text{buy}(\text{wine}); \text{buy}(\text{corn}) \} :- \text{at}(\text{grocery})$.
- Another example $P = \{ \{a\} \leftarrow b, b \leftarrow \}$ has two stable models: $\{b\}$ and $\{a, b\}$

Embedding in normal rules

- A choice rule of form

$$\{a_1, \dots, a_m\} \leftarrow a_{m+1}, \dots, a_n, \sim a_{n+1}, \dots, \sim a_o$$

can be translated into $2m + 1$ normal rules

$$\begin{array}{l} b \leftarrow a_{m+1}, \dots, a_n, \sim a_{n+1}, \dots, \sim a_o \\ a_1 \leftarrow b, \sim a'_1 \quad \dots \quad a_m \leftarrow b, \sim a'_m \\ a'_1 \leftarrow \sim a_1 \quad \dots \quad a'_m \leftarrow \sim a_m \end{array}$$

by introducing new atoms b, a'_1, \dots, a'_m .

Embedding in normal rules

- A choice rule of form

$$\{a_1, \dots, a_m\} \leftarrow a_{m+1}, \dots, a_n, \sim a_{n+1}, \dots, \sim a_o$$

can be translated into $2m + 1$ normal rules

$$\begin{array}{l} b \leftarrow a_{m+1}, \dots, a_n, \sim a_{n+1}, \dots, \sim a_o \\ a_1 \leftarrow b, \sim a'_1 \quad \dots \quad a_m \leftarrow b, \sim a'_m \\ a'_1 \leftarrow \sim a_1 \quad \dots \quad a'_m \leftarrow \sim a_m \end{array}$$

by introducing new atoms b, a'_1, \dots, a'_m .

Embedding in normal rules

- A choice rule of form

$$\{a_1, \dots, a_m\} \leftarrow a_{m+1}, \dots, a_n, \sim a_{n+1}, \dots, \sim a_o$$

can be translated into $2m + 1$ normal rules

$$\begin{array}{l} b \leftarrow a_{m+1}, \dots, a_n, \sim a_{n+1}, \dots, \sim a_o \\ a_1 \leftarrow b, \sim a'_1 \quad \dots \quad a_m \leftarrow b, \sim a'_m \\ a'_1 \leftarrow \sim a_1 \quad \dots \quad a'_m \leftarrow \sim a_m \end{array}$$

by introducing new atoms b, a'_1, \dots, a'_m .

Outline

- 1 Motivation
- 2 Core language
 - Integrity constraint
 - Choice rule
 - Cardinality rule
 - Weight rule
- 3 Extended language
- 4 Intermediate formats

Cardinality rule

- Idea Control (lower) cardinality of subsets
- Syntax A **cardinality rule** is the form

$$a_0 \leftarrow l \{ a_1, \dots, a_m, \sim a_{m+1}, \dots, \sim a_n \}$$

where $0 \leq m \leq n$ and each a_i is an atom for $1 \leq i \leq n$;
 l is a non-negative integer.

- Informal meaning The head atom belongs to the stable model, if at least l elements of the body are included in the stable model
- Note l acts as a lower bound on the body
- Example `pass(c42) :- 2 { pass(a1); pass(a2); pass(a3) }.`
- Another example $P = \{a \leftarrow 1\{b, c\}, b \leftarrow\}$ has stable model $\{a, b\}$

Cardinality rule

- Idea Control (lower) cardinality of subsets
- Syntax A **cardinality rule** is the form

$$a_0 \leftarrow l \{ a_1, \dots, a_m, \sim a_{m+1}, \dots, \sim a_n \}$$

where $0 \leq m \leq n$ and each a_i is an atom for $1 \leq i \leq n$;
 l is a non-negative integer.

- Informal meaning The head atom belongs to the stable model, if at least l elements of the body are included in the stable model
- Note l acts as a **lower bound** on the body
- Example `pass(c42) :- 2 { pass(a1); pass(a2); pass(a3) }.`
- Another example `P = {a ← 1{b, c}, b ←}` has stable model `{a, b}`

Cardinality rule

- Idea Control (lower) cardinality of subsets
- Syntax A **cardinality rule** is the form

$$a_0 \leftarrow l \{ a_1, \dots, a_m, \sim a_{m+1}, \dots, \sim a_n \}$$

where $0 \leq m \leq n$ and each a_i is an atom for $1 \leq i \leq n$;
 l is a non-negative integer.

- Informal meaning The head atom belongs to the stable model, if at least l elements of the body are included in the stable model
- Note l acts as a **lower bound** on the body
- Example `pass(c42) :- 2 { pass(a1); pass(a2); pass(a3) }.`
- Another example $P = \{a \leftarrow 1\{b, c\}, b \leftarrow\}$ has stable model $\{a, b\}$

Cardinality rule

- Idea Control (lower) cardinality of subsets
- Syntax A **cardinality rule** is the form

$$a_0 \leftarrow l \{ a_1, \dots, a_m, \sim a_{m+1}, \dots, \sim a_n \}$$

where $0 \leq m \leq n$ and each a_i is an atom for $1 \leq i \leq n$;
 l is a non-negative integer.

- Informal meaning The head atom belongs to the stable model, if at least l elements of the body are included in the stable model
- Note l acts as a **lower bound** on the body
- Example `pass(c42) :- 2 { pass(a1); pass(a2); pass(a3) }.`
- Another example $P = \{a \leftarrow 1\{b, c\}, b \leftarrow\}$ has stable model $\{a, b\}$

Embedding in normal rules

- Replace each cardinality rule

$$a_0 \leftarrow l \{ a_1, \dots, a_m, \sim a_{m+1}, \dots, \sim a_n \}$$

by $a_0 \leftarrow ctr(1, l)$

where atom $ctr(i, j)$ represents the fact that at least j of the literals having an equal or greater index than i , are in a stable model

- The definition of $ctr/2$ is given for $0 \leq k \leq l$ by the rules

$$\begin{aligned} ctr(i, k+1) &\leftarrow ctr(i+1, k), a_i \\ ctr(i, k) &\leftarrow ctr(i+1, k) \end{aligned} \quad \text{for } 1 \leq i \leq m$$

$$\begin{aligned} ctr(j, k+1) &\leftarrow ctr(j+1, k), \sim a_j \\ ctr(j, k) &\leftarrow ctr(j+1, k) \end{aligned} \quad \text{for } m+1 \leq j \leq n$$

$$ctr(n+1, 0) \leftarrow$$

Embedding in normal rules

- Replace each cardinality rule

$$a_0 \leftarrow l \{ a_1, \dots, a_m, \sim a_{m+1}, \dots, \sim a_n \}$$

by $a_0 \leftarrow ctr(1, l)$

where atom $ctr(i, j)$ represents the fact that at least j of the literals having an equal or greater index than i , are in a stable model

- The definition of $ctr/2$ is given for $0 \leq k \leq l$ by the rules

$$\begin{aligned} ctr(i, k+1) &\leftarrow ctr(i+1, k), a_i \\ ctr(i, k) &\leftarrow ctr(i+1, k) \end{aligned} \quad \text{for } 1 \leq i \leq m$$

$$\begin{aligned} ctr(j, k+1) &\leftarrow ctr(j+1, k), \sim a_j \\ ctr(j, k) &\leftarrow ctr(j+1, k) \end{aligned} \quad \text{for } m+1 \leq j \leq n$$

$$ctr(n+1, 0) \leftarrow$$

Embedding in normal rules

- Replace each cardinality rule

$$a_0 \leftarrow l \{ a_1, \dots, a_m, \sim a_{m+1}, \dots, \sim a_n \}$$

by $a_0 \leftarrow ctr(1, l)$

where atom $ctr(i, j)$ represents the fact that at least j of the literals having an equal or greater index than i , are in a stable model

- The definition of $ctr/2$ is given for $0 \leq k \leq l$ by the rules

$$\begin{aligned} ctr(i, k+1) &\leftarrow ctr(i+1, k), a_i \\ ctr(i, k) &\leftarrow ctr(i+1, k) \end{aligned} \quad \text{for } 1 \leq i \leq m$$

$$\begin{aligned} ctr(j, k+1) &\leftarrow ctr(j+1, k), \sim a_j \\ ctr(j, k) &\leftarrow ctr(j+1, k) \end{aligned} \quad \text{for } m+1 \leq j \leq n$$

$$ctr(n+1, 0) \leftarrow$$

Embedding in normal rules

- Replace each cardinality rule

$$a_0 \leftarrow l \{ a_1, \dots, a_m, \sim a_{m+1}, \dots, \sim a_n \}$$

by $a_0 \leftarrow ctr(1, l)$

where atom $ctr(i, j)$ represents the fact that at least j of the literals having an equal or greater index than i , are in a stable model

- The definition of $ctr/2$ is given for $0 \leq k \leq l$ by the rules

$$\begin{aligned} ctr(i, k+1) &\leftarrow ctr(i+1, k), a_i \\ ctr(i, k) &\leftarrow ctr(i+1, k) \end{aligned} \quad \text{for } 1 \leq i \leq m$$

$$\begin{aligned} ctr(j, k+1) &\leftarrow ctr(j+1, k), \sim a_j \\ ctr(j, k) &\leftarrow ctr(j+1, k) \end{aligned} \quad \text{for } m+1 \leq j \leq n$$

$$ctr(n+1, 0) \leftarrow$$

Embedding in normal rules

- Replace each cardinality rule

$$a_0 \leftarrow l \{ a_1, \dots, a_m, \sim a_{m+1}, \dots, \sim a_n \}$$

by $a_0 \leftarrow ctr(1, l)$

where atom $ctr(i, j)$ represents the fact that at least j of the literals having an equal or greater index than i , are in a stable model

- The definition of $ctr/2$ is given for $0 \leq k \leq l$ by the rules

$$\begin{aligned} ctr(i, k+1) &\leftarrow ctr(i+1, k), a_i \\ ctr(i, k) &\leftarrow ctr(i+1, k) \end{aligned} \quad \text{for } 1 \leq i \leq m$$

$$\begin{aligned} ctr(j, k+1) &\leftarrow ctr(j+1, k), \sim a_j \\ ctr(j, k) &\leftarrow ctr(j+1, k) \end{aligned} \quad \text{for } m+1 \leq j \leq n$$

$$ctr(n+1, 0) \leftarrow$$

Embedding in normal rules

- Replace each cardinality rule

$$a_0 \leftarrow l \{ a_1, \dots, a_m, \sim a_{m+1}, \dots, \sim a_n \}$$

by $a_0 \leftarrow ctr(1, l)$

where atom $ctr(i, j)$ represents the fact that at least j of the literals having an equal or greater index than i , are in a stable model

- The definition of $ctr/2$ is given for $0 \leq k \leq l$ by the rules

$$\begin{aligned} ctr(i, k+1) &\leftarrow ctr(i+1, k), a_i \\ ctr(i, k) &\leftarrow ctr(i+1, k) \end{aligned} \quad \text{for } 1 \leq i \leq m$$

$$\begin{aligned} ctr(j, k+1) &\leftarrow ctr(j+1, k), \sim a_j \\ ctr(j, k) &\leftarrow ctr(j+1, k) \end{aligned} \quad \text{for } m+1 \leq j \leq n$$

$$ctr(n+1, 0) \leftarrow$$

Embedding in normal rules

- Replace each cardinality rule

$$a_0 \leftarrow l \{ a_1, \dots, a_m, \sim a_{m+1}, \dots, \sim a_n \}$$

by $a_0 \leftarrow ctr(1, l)$

where atom $ctr(i, j)$ represents the fact that at least j of the literals having an equal or greater index than i , are in a stable model

- The definition of $ctr/2$ is given for $0 \leq k \leq l$ by the rules

$$\begin{aligned} ctr(i, k+1) &\leftarrow ctr(i+1, k), a_i \\ ctr(i, k) &\leftarrow ctr(i+1, k) \end{aligned} \quad \text{for } 1 \leq i \leq m$$

$$\begin{aligned} ctr(j, k+1) &\leftarrow ctr(j+1, k), \sim a_j \\ ctr(j, k) &\leftarrow ctr(j+1, k) \end{aligned} \quad \text{for } m+1 \leq j \leq n$$

$$ctr(n+1, 0) \leftarrow$$

An example

- Program $\{a \leftarrow, c \leftarrow 1 \{a, b\}\}$ has the stable model $\{a, c\}$
- Translating the cardinality rule yields the rules

$$\begin{array}{l}
 a \leftarrow \\
 c \leftarrow ctr(1, 1) \\
 ctr(1, 2) \leftarrow ctr(2, 1), a \\
 ctr(1, 1) \leftarrow ctr(2, 1) \\
 ctr(2, 2) \leftarrow ctr(3, 1), b \\
 ctr(2, 1) \leftarrow ctr(3, 1) \\
 ctr(1, 1) \leftarrow ctr(2, 0), a \\
 ctr(1, 0) \leftarrow ctr(2, 0) \\
 ctr(2, 1) \leftarrow ctr(3, 0), b \\
 ctr(2, 0) \leftarrow ctr(3, 0) \\
 ctr(3, 0) \leftarrow
 \end{array}$$

having stable model $\{a, ctr(3, 0), ctr(2, 0), ctr(1, 0), ctr(1, 1), c\}$

An example

- Program $\{a \leftarrow, c \leftarrow 1 \{a, b\}\}$ has the stable model $\{a, c\}$
- Translating the cardinality rule yields the rules

$$\begin{array}{l}
 a \leftarrow \\
 c \leftarrow ctr(1, 1) \\
 ctr(1, 2) \leftarrow ctr(2, 1), a \\
 ctr(1, 1) \leftarrow ctr(2, 1) \\
 ctr(2, 2) \leftarrow ctr(3, 1), b \\
 ctr(2, 1) \leftarrow ctr(3, 1) \\
 ctr(1, 1) \leftarrow ctr(2, 0), a \\
 ctr(1, 0) \leftarrow ctr(2, 0) \\
 ctr(2, 1) \leftarrow ctr(3, 0), b \\
 ctr(2, 0) \leftarrow ctr(3, 0) \\
 ctr(3, 0) \leftarrow
 \end{array}$$

having stable model $\{a, ctr(3, 0), ctr(2, 0), ctr(1, 0), ctr(1, 1), c\}$

... and vice versa

- A normal rule

$$a_0 \leftarrow a_1, \dots, a_m, \sim a_{m+1}, \dots, \sim a_n$$

can be represented by the cardinality rule

$$a_0 \leftarrow n \{a_1, \dots, a_m, \sim a_{m+1}, \dots, \sim a_n\}$$

Cardinality rules with upper bounds

- A rule of the form

$$a_0 \leftarrow l \{ a_1, \dots, a_m, \sim a_{m+1}, \dots, \sim a_n \} u \quad (1)$$

where $0 \leq m \leq n$ and each a_i is an atom for $1 \leq i \leq n$;
 l and u are non-negative integers

stands for

$$\begin{aligned} a_0 &\leftarrow b, \sim c \\ b &\leftarrow l \{ a_1, \dots, a_m, \sim a_{m+1}, \dots, \sim a_n \} \\ c &\leftarrow u+1 \{ a_1, \dots, a_m, \sim a_{m+1}, \dots, \sim a_n \} \end{aligned}$$

where b and c are new symbols

- Note The single constraint in the body of the cardinality rule (1) is referred to as a cardinality constraint

Cardinality rules with upper bounds

- A rule of the form

$$a_0 \leftarrow l \{ a_1, \dots, a_m, \sim a_{m+1}, \dots, \sim a_n \} u \quad (1)$$

where $0 \leq m \leq n$ and each a_i is an atom for $1 \leq i \leq n$;
 l and u are non-negative integers

stands for

$$\begin{aligned} a_0 &\leftarrow b, \sim c \\ b &\leftarrow l \{ a_1, \dots, a_m, \sim a_{m+1}, \dots, \sim a_n \} \\ c &\leftarrow u+1 \{ a_1, \dots, a_m, \sim a_{m+1}, \dots, \sim a_n \} \end{aligned}$$

where b and c are new symbols

- Note The single constraint in the body of the cardinality rule (1) is referred to as a cardinality constraint

Cardinality rules with upper bounds

- A rule of the form

$$a_0 \leftarrow l \{ a_1, \dots, a_m, \sim a_{m+1}, \dots, \sim a_n \} u \quad (1)$$

where $0 \leq m \leq n$ and each a_i is an atom for $1 \leq i \leq n$;
 l and u are non-negative integers

stands for

$$\begin{aligned} a_0 &\leftarrow b, \sim c \\ b &\leftarrow l \{ a_1, \dots, a_m, \sim a_{m+1}, \dots, \sim a_n \} \\ c &\leftarrow u+1 \{ a_1, \dots, a_m, \sim a_{m+1}, \dots, \sim a_n \} \end{aligned}$$

where b and c are new symbols

- Note The single constraint in the body of the cardinality rule (1) is referred to as a **cardinality constraint**

Cardinality constraints

- Syntax A **cardinality constraint** is of the form

$$l \{ a_1, \dots, a_m, \sim a_{m+1}, \dots, \sim a_n \} u$$

where $0 \leq m \leq n$ and each a_i is an atom for $1 \leq i \leq n$;
 l and u are non-negative integers

- Informal meaning A cardinality constraint is satisfied by a stable model X , if the number of its contained literals satisfied by X is between l and u (inclusive)
- In other words, if

$$l \leq |(\{a_1, \dots, a_m\} \cap X) \cup (\{a_{m+1}, \dots, a_n\} \setminus X)| \leq u$$

Cardinality constraints

- Syntax A **cardinality constraint** is of the form

$$l \{ a_1, \dots, a_m, \sim a_{m+1}, \dots, \sim a_n \} u$$

where $0 \leq m \leq n$ and each a_i is an atom for $1 \leq i \leq n$;
 l and u are non-negative integers

- Informal meaning A cardinality constraint is satisfied by a stable model X , if the number of its contained literals satisfied by X is between l and u (inclusive)
- In other words, if

$$l \leq |(\{a_1, \dots, a_m\} \cap X) \cup (\{a_{m+1}, \dots, a_n\} \setminus X)| \leq u$$

Cardinality constraints

- Syntax A **cardinality constraint** is of the form

$$l \{ a_1, \dots, a_m, \sim a_{m+1}, \dots, \sim a_n \} u$$

where $0 \leq m \leq n$ and each a_i is an atom for $1 \leq i \leq n$;
 l and u are non-negative integers

- Informal meaning A cardinality constraint is satisfied by a stable model X , if the number of its contained literals satisfied by X is between l and u (inclusive)
- In other words, if

$$l \leq |(\{a_1, \dots, a_m\} \cap X) \cup (\{a_{m+1}, \dots, a_n\} \setminus X)| \leq u$$

Cardinality constraints as heads

- A rule of the form

$$l \{a_1, \dots, a_m, \sim a_{m+1}, \dots, \sim a_n\} u \leftarrow a_{n+1}, \dots, a_o, \sim a_{o+1}, \dots, \sim a_p$$

where $0 \leq m \leq n \leq o \leq p$ and each a_i is an atom for $1 \leq i \leq p$;
 l and u are non-negative integers

stands for

$$\begin{aligned} b &\leftarrow a_{n+1}, \dots, a_o, \sim a_{o+1}, \dots, \sim a_p \\ \{a_1, \dots, a_m\} &\leftarrow b \\ c &\leftarrow l \{a_1, \dots, a_m, \sim a_{m+1}, \dots, \sim a_n\} u \\ &\leftarrow b, \sim c \end{aligned}$$

where b and c are new symbols

- Example $1\{ \text{color}(v42, \text{red}); \text{color}(v42, \text{green}); \text{color}(v42, \text{blue}) \} 1.$

Cardinality constraints as heads

- A rule of the form

$$l \{a_1, \dots, a_m, \sim a_{m+1}, \dots, \sim a_n\} u \leftarrow a_{n+1}, \dots, a_o, \sim a_{o+1}, \dots, \sim a_p$$

where $0 \leq m \leq n \leq o \leq p$ and each a_i is an atom for $1 \leq i \leq p$;
 l and u are non-negative integers

stands for

$$\begin{aligned} b &\leftarrow a_{n+1}, \dots, a_o, \sim a_{o+1}, \dots, \sim a_p \\ \{a_1, \dots, a_m\} &\leftarrow b \\ c &\leftarrow l \{a_1, \dots, a_m, \sim a_{m+1}, \dots, \sim a_n\} u \\ &\leftarrow b, \sim c \end{aligned}$$

where b and c are new symbols

- Example `1{ color(v42,red); color(v42,green); color(v42,blue) }1.`

Cardinality constraints as heads

- A rule of the form

$$l \{a_1, \dots, a_m, \sim a_{m+1}, \dots, \sim a_n\} u \leftarrow a_{n+1}, \dots, a_o, \sim a_{o+1}, \dots, \sim a_p$$

where $0 \leq m \leq n \leq o \leq p$ and each a_i is an atom for $1 \leq i \leq p$;
 l and u are non-negative integers

stands for

$$\begin{aligned} b &\leftarrow a_{n+1}, \dots, a_o, \sim a_{o+1}, \dots, \sim a_p \\ \{a_1, \dots, a_m\} &\leftarrow b \\ c &\leftarrow l \{a_1, \dots, a_m, \sim a_{m+1}, \dots, \sim a_n\} u \\ &\leftarrow b, \sim c \end{aligned}$$

where b and c are new symbols

- Example $1\{ \text{color}(v42, \text{red}); \text{color}(v42, \text{green}); \text{color}(v42, \text{blue}) \} 1.$

Full-fledged cardinality rules

- A rule of the form

$$l_0 S_0 u_0 \leftarrow l_1 S_1 u_1, \dots, l_n S_n u_n$$

where each $l_i S_i u_i$ is a cardinality constraint for $0 \leq i \leq n$
stands for

$$a \leftarrow b_1, \dots, b_n, \sim c_1, \dots, \sim c_n$$

$$S_0^+ \leftarrow a$$

$$\leftarrow a, \sim b_0$$

$$\leftarrow a, c_0$$

$$b_i \leftarrow l_i S_i$$

$$c_i \leftarrow u_i + 1 S_i$$

where a, b_i, c_i are new symbols (and \cdot^+ is defined as on Slide ??)

Full-fledged cardinality rules

- A rule of the form

$$l_0 S_0 u_0 \leftarrow l_1 S_1 u_1, \dots, l_n S_n u_n$$

where each $l_i S_i u_i$ is a cardinality constraint for $0 \leq i \leq n$
stands for

$$a \leftarrow b_1, \dots, b_n, \sim c_1, \dots, \sim c_n$$

$$S_0^+ \leftarrow a$$

$$\leftarrow a, \sim b_0$$

$$\leftarrow a, c_0$$

$$b_i \leftarrow l_i S_i$$

$$c_i \leftarrow u_i + 1 S_i$$

where a, b_i, c_i are new symbols (and \cdot^+ is defined as on Slide ??)

Full-fledged cardinality rules

- A rule of the form

$$l_0 S_0 u_0 \leftarrow l_1 S_1 u_1, \dots, l_n S_n u_n$$

where each $l_i S_i u_i$ is a cardinality constraint for $0 \leq i \leq n$
stands for

$$a \leftarrow b_1, \dots, b_n, \sim c_1, \dots, \sim c_n$$

$$S_0^+ \leftarrow a$$

$$\leftarrow a, \sim b_0$$

$$\leftarrow a, c_0$$

$$b_i \leftarrow l_i S_i$$

$$c_i \leftarrow u_i + 1 S_i$$

where a, b_i, c_i are new symbols (and \cdot^+ is defined as on Slide ??)

Full-fledged cardinality rules

- A rule of the form

$$l_0 S_0 u_0 \leftarrow l_1 S_1 u_1, \dots, l_n S_n u_n$$

where each $l_i S_i u_i$ is a cardinality constraint for $0 \leq i \leq n$
stands for

$$a \leftarrow b_1, \dots, b_n, \sim c_1, \dots, \sim c_n$$

$$S_0^+ \leftarrow a$$

$$\leftarrow a, \sim b_0$$

$$\leftarrow a, c_0$$

$$b_i \leftarrow l_i S_i$$

$$c_i \leftarrow u_i + 1 S_i$$

where a, b_i, c_i are new symbols (and \cdot^+ is defined as on Slide ??)

Full-fledged cardinality rules

- A rule of the form

$$l_0 S_0 u_0 \leftarrow l_1 S_1 u_1, \dots, l_n S_n u_n$$

where each $l_i S_i u_i$ is a cardinality constraint for $0 \leq i \leq n$
stands for

$$a \leftarrow b_1, \dots, b_n, \sim c_1, \dots, \sim c_n$$

$$S_0^+ \leftarrow a$$

$$\leftarrow a, \sim b_0$$

$$\leftarrow a, c_0$$

$$b_i \leftarrow l_i S_i$$

$$c_i \leftarrow u_{i+1} S_i$$

where a, b_i, c_i are new symbols (and \cdot^+ is defined as on Slide ??)

Full-fledged cardinality rules

- A rule of the form

$$l_0 S_0 u_0 \leftarrow l_1 S_1 u_1, \dots, l_n S_n u_n$$

where each $l_i S_i u_i$ is a cardinality constraint for $0 \leq i \leq n$
stands for

$$a \leftarrow b_1, \dots, b_n, \sim c_1, \dots, \sim c_n$$

$$S_0^+ \leftarrow a$$

$$\leftarrow a, \sim b_0$$

$$\leftarrow a, c_0$$

$$b_i \leftarrow l_i S_i$$

$$c_i \leftarrow u_i + 1 S_i$$

where a, b_i, c_i are new symbols (and \cdot^+ is defined as on Slide ??)

Outline

- 1 Motivation
- 2 Core language
 - Integrity constraint
 - Choice rule
 - Cardinality rule
 - Weight rule
- 3 Extended language
- 4 Intermediate formats

Weight rule

- Syntax A **weight rule** is the form

$$a_0 \leftarrow l \{ w_1 : a_1, \dots, w_m : a_m, w_{m+1} : \sim a_{m+1}, \dots, w_n : \sim a_n \}$$

where $0 \leq m \leq n$ and each a_i is an atom;

l and w_i are integers for $1 \leq i \leq n$

- A weighted literal $w_i : l_i$ associates each literal l_i with a weight w_i
- Note A cardinality rule is a weight rule where $w_i = 1$ for $0 \leq i \leq n$

Weight rule

- Syntax A **weight rule** is the form

$$a_0 \leftarrow l \{ w_1 : a_1, \dots, w_m : a_m, w_{m+1} : \sim a_{m+1}, \dots, w_n : \sim a_n \}$$

where $0 \leq m \leq n$ and each a_i is an atom;

l and w_i are integers for $1 \leq i \leq n$

- A weighted literal $w_i : l_i$ associates each literal l_i with a weight w_i
- Note A cardinality rule is a weight rule where $w_i = 1$ for $0 \leq i \leq n$

Weight constraints

- Syntax A **weight constraint** is of the form

$$l \{ w_1 : a_1, \dots, w_m : a_m, w_{m+1} : \sim a_{m+1}, \dots, w_n : \sim a_n \} u$$

where $0 \leq m \leq n$ and each a_i is an atom;

l, u and w_i are integers for $1 \leq i \leq n$

- Meaning A weight constraint is satisfied by a stable model X , if

$$l \leq \left(\sum_{1 \leq i \leq m, a_i \in X} w_i + \sum_{m < i \leq n, a_i \notin X} w_i \right) \leq u$$

- Note (Cardinality and) weight constraints amount to constraints on (count and) sum aggregate functions
- Example

$10 \{ 4:\text{course}(\text{db}); 6:\text{course}(\text{ai}); 8:\text{course}(\text{project}); 3:\text{course}(\text{xml}) \} 20$

Weight constraints

- Syntax A **weight constraint** is of the form

$$l \{ w_1 : a_1, \dots, w_m : a_m, w_{m+1} : \sim a_{m+1}, \dots, w_n : \sim a_n \} u$$

where $0 \leq m \leq n$ and each a_i is an atom;

l, u and w_i are integers for $1 \leq i \leq n$

- Meaning A weight constraint is satisfied by a stable model X , if

$$l \leq \left(\sum_{1 \leq i \leq m, a_i \in X} w_i + \sum_{m < i \leq n, a_i \notin X} w_i \right) \leq u$$

- Note (Cardinality and) weight constraints amount to constraints on (count and) sum aggregate functions
- Example

$10 \{ 4:\text{course}(\text{db}); 6:\text{course}(\text{ai}); 8:\text{course}(\text{project}); 3:\text{course}(\text{xml}) \} 20$

Weight constraints

- Syntax A **weight constraint** is of the form

$$l \{ w_1 : a_1, \dots, w_m : a_m, w_{m+1} : \sim a_{m+1}, \dots, w_n : \sim a_n \} u$$

where $0 \leq m \leq n$ and each a_i is an atom;

l, u and w_i are integers for $1 \leq i \leq n$

- Meaning A weight constraint is satisfied by a stable model X , if

$$l \leq \left(\sum_{1 \leq i \leq m, a_i \in X} w_i + \sum_{m < i \leq n, a_i \notin X} w_i \right) \leq u$$

- Note (Cardinality and) weight constraints amount to constraints on (count and) sum aggregate functions

- Example

$10 \{ 4 : \text{course}(\text{db}); 6 : \text{course}(\text{ai}); 8 : \text{course}(\text{project}); 3 : \text{course}(\text{xml}) \} 20$

Weight constraints

- Syntax A **weight constraint** is of the form

$$l \{ w_1 : a_1, \dots, w_m : a_m, w_{m+1} : \sim a_{m+1}, \dots, w_n : \sim a_n \} u$$

where $0 \leq m \leq n$ and each a_i is an atom;

l, u and w_i are integers for $1 \leq i \leq n$

- Meaning A weight constraint is satisfied by a stable model X , if

$$l \leq \left(\sum_{1 \leq i \leq m, a_i \in X} w_i + \sum_{m < i \leq n, a_i \notin X} w_i \right) \leq u$$

- Note (Cardinality and) weight constraints amount to constraints on (count and) sum aggregate functions
- Example

$10 \{ 4:\text{course}(\text{db}); 6:\text{course}(\text{ai}); 8:\text{course}(\text{project}); 3:\text{course}(\text{xml}) \} 20$

Outline

- 1 Motivation
- 2 Core language
- 3 Extended language
- 4 Intermediate formats

Outline

- 1 Motivation
- 2 Core language
- 3 Extended language
 - Conditional literal
 - Optimization statement
- 4 Intermediate formats

Conditional literals

- Syntax A **conditional literal** is of the form

$$l : l_1, \dots, l_n$$

where l and l_i are literals for $0 \leq i \leq n$

- Informal meaning A conditional literal can be regarded as the list of elements in the set $\{l \mid l_1, \dots, l_n\}$
- Note The expansion of conditional literals is context dependent
- Example Given 'p(1..3). q(2).'

$r(X) : p(X), \text{not } q(X) :- r(X) : p(X), \text{not } q(X); 1\{r(X) : p(X), \text{not } q(X)\}.$

is instantiated to

$r(1); r(3) :- r(1), r(3), 1\{r(1); r(3)\}.$

Conditional literals

- Syntax A **conditional literal** is of the form

$$l : l_1, \dots, l_n$$

where l and l_i are literals for $0 \leq i \leq n$

- Informal meaning A conditional literal can be regarded as the list of elements in the set $\{l \mid l_1, \dots, l_n\}$
- Note The expansion of conditional literals is context dependent
- Example Given 'p(1..3). q(2).'

$r(X) : p(X), \text{not } q(X) :- r(X) : p(X), \text{not } q(X); 1\{r(X) : p(X), \text{not } q(X)\}.$

is instantiated to

$r(1); r(3) :- r(1), r(3), 1\{r(1); r(3)\}.$

Conditional literals

- Syntax A **conditional literal** is of the form

$$l : l_1, \dots, l_n$$

where l and l_i are literals for $0 \leq i \leq n$

- Informal meaning A conditional literal can be regarded as the list of elements in the set $\{l \mid l_1, \dots, l_n\}$
- Note The expansion of conditional literals is context dependent
- Example Given 'p(1..3). q(2).'

$r(X) : p(X), \text{not } q(X) :- r(X) : p(X), \text{not } q(X); 1 \{ r(X) : p(X), \text{not } q(X) \}.$

is instantiated to

$r(1); r(3) :- r(1), r(3), 1 \{ r(1); r(3) \}.$

Conditional literals

- Syntax A **conditional literal** is of the form

$$l : l_1, \dots, l_n$$

where l and l_i are literals for $0 \leq i \leq n$

- Informal meaning A conditional literal can be regarded as the list of elements in the set $\{l \mid l_1, \dots, l_n\}$
- Note The expansion of conditional literals is context dependent
- Example Given 'p(1..3). q(2).'

$r(X) : p(X), \text{not } q(X) :- r(X) : p(X), \text{not } q(X); 1\{r(X) : p(X), \text{not } q(X)\}.$

is instantiated to

$r(1); r(3) :- r(1), r(3), 1\{r(1); r(3)\}.$

Conditional literals

- Syntax A **conditional literal** is of the form

$$l : l_1, \dots, l_n$$

where l and l_i are literals for $0 \leq i \leq n$

- Informal meaning A conditional literal can be regarded as the list of elements in the set $\{l \mid l_1, \dots, l_n\}$
- Note The expansion of conditional literals is context dependent
- Example Given 'p(1..3). q(2).'

$r(X) : p(X), \text{not } q(X) :- r(X) : p(X), \text{not } q(X); 1 \{ r(X) : p(X), \text{not } q(X) \}.$

is instantiated to

$r(1); r(3) :- r(1), r(3), 1 \{ r(1); r(3) \}.$

Conditional literals

- Syntax A **conditional literal** is of the form

$$l : l_1, \dots, l_n$$

where l and l_i are literals for $0 \leq i \leq n$

- Informal meaning A conditional literal can be regarded as the list of elements in the set $\{l \mid l_1, \dots, l_n\}$
- Note The expansion of conditional literals is context dependent
- Example Given 'p(1..3). q(2).'

$r(X) : p(X), \text{not } q(X) \text{ :- } r(X) : p(X), \text{not } q(X); 1 \{ r(X) : p(X), \text{not } q(X) \}.$

is instantiated to

$r(1); r(3) \text{ :- } r(1), r(3), 1 \{ r(1); r(3) \}.$

Conditional literals

- Syntax A **conditional literal** is of the form

$$l : l_1, \dots, l_n$$

where l and l_i are literals for $0 \leq i \leq n$

- Informal meaning A conditional literal can be regarded as the list of elements in the set $\{l \mid l_1, \dots, l_n\}$
- Note The expansion of conditional literals is context dependent
- Example Given 'p(1..3). q(2).'

$r(X) : p(X), \text{not } q(X) :- r(X) : p(X), \text{not } q(X); 1 \{ r(X) : p(X), \text{not } q(X) \}.$

is instantiated to

$r(1); r(3) :- r(1), r(3), 1 \{ r(1); r(3) \}.$

Outline

- 1 Motivation
- 2 Core language
- 3 Extended language
 - Conditional literal
 - Optimization statement
- 4 Intermediate formats

Optimization statement

- Idea Express (multiple) cost functions subject to minimization and/or maximization
- Syntax A **minimize statement** is of the form

$$\textit{minimize} \{ w_1 @ p_1 : l_{1_1}, \dots, l_{m_1}; \dots; w_n @ p_n : l_{1_n}, \dots, l_{m_n} \}.$$

where each l_{j_i} is a literal; and w_i and p_i are integers for $1 \leq i \leq n$

Priority levels, p_i , allow for representing lexicographically ordered minimization objectives

- Meaning A minimize statement is a directive that instructs the ASP solver to compute optimal stable models by minimizing a weighted sum of elements

Optimization statement

- Idea Express (multiple) cost functions subject to minimization and/or maximization
- Syntax A **minimize statement** is of the form

$$\textit{minimize} \{ w_1 @ p_1 : l_{1_1}, \dots, l_{m_1}; \dots; w_n @ p_n : l_{1_n}, \dots, l_{m_n} \}.$$

where each l_{j_i} is a literal; and w_i and p_i are integers for $1 \leq i \leq n$

Priority levels, p_i , allow for representing lexicographically ordered minimization objectives

- Meaning A minimize statement is a directive that instructs the ASP solver to compute optimal stable models by minimizing a weighted sum of elements

Optimization statement

- Idea Express (multiple) cost functions subject to minimization and/or maximization
- Syntax A **minimize statement** is of the form

$$\textit{minimize} \{ w_1 @ p_1 : l_{1_1}, \dots, l_{m_1}; \dots; w_n @ p_n : l_{1_n}, \dots, l_{m_n} \}.$$

where each l_{j_i} is a literal; and w_i and p_i are integers for $1 \leq i \leq n$

Priority levels, p_i , allow for representing lexicographically ordered minimization objectives

- Meaning A minimize statement is a directive that instructs the ASP solver to compute optimal stable models by minimizing a weighted sum of elements

Optimization statement

- A maximize statement of the form

$$\textit{maximize} \{ w_1 @ p_1 : l_1, \dots, w_n @ p_n : l_n \}$$

stands for $\textit{minimize} \{ -w_1 @ p_1 : l_1, \dots, -w_n @ p_n : l_n \}$

- Example When configuring a computer, we may want to maximize hard disk capacity, while minimizing price

```
#maximize { 250@1:hd(1); 500@1:hd(2); 750@1:hd(3); 1000@1:hd(4) }.
#minimize { 30@2:hd(1); 40@2:hd(2); 60@2:hd(3); 80@2:hd(4) }.
```

The priority levels indicate that (minimizing) price is more important than (maximizing) capacity

Optimization statement

- A maximize statement of the form

$$\text{maximize } \{ w_1 @ p_1 : l_1, \dots, w_n @ p_n : l_n \}$$

stands for $\text{minimize } \{ -w_1 @ p_1 : l_1, \dots, -w_n @ p_n : l_n \}$

- Example When configuring a computer, we may want to maximize hard disk capacity, while minimizing price

```
#maximize { 250@1:hd(1); 500@1:hd(2); 750@1:hd(3); 1000@1:hd(4) }.
#minimize { 30@2:hd(1); 40@2:hd(2); 60@2:hd(3); 80@2:hd(4) }.
```

The priority levels indicate that (minimizing) price is more important than (maximizing) capacity

Optimization statement

- A maximize statement of the form

$$\text{maximize } \{ w_1 @ p_1 : l_1, \dots, w_n @ p_n : l_n \}$$

stands for $\text{minimize } \{ -w_1 @ p_1 : l_1, \dots, -w_n @ p_n : l_n \}$

- Example When configuring a computer, we may want to maximize hard disk capacity, while minimizing price

```
#maximize { P@1:hd(I,P,C) }.  
#minimize { C@2:hd(I,P,C) }.
```

The priority levels indicate that (minimizing) price is more important than (maximizing) capacity

Weak constraints

- Weak constraints are an alternative to minimize statements
- Syntax $:\sim l_1, \dots, l_n [w@l]$
where each l_i is a literal; and w_i and p_i are integers for $1 \leq i \leq n$
- Example
 - $:\sim \text{hd}(1) . [30@2]$
 - $:\sim \text{hd}(2) . [40@2]$
 - $:\sim \text{hd}(3) . [60@2]$
 - $:\sim \text{hd}(4) . [80@2]$

Weak constraints

- Weak constraints are an alternative to minimize statements
- Syntax $:\sim l_1, \dots, l_n [w@l]$
where each l_i is a literal; and w_i and p_i are integers for $1 \leq i \leq n$
- Example
 - $:\sim \text{hd}(1) . [30@2]$
 - $:\sim \text{hd}(2) . [40@2]$
 - $:\sim \text{hd}(3) . [60@2]$
 - $:\sim \text{hd}(4) . [80@2]$

Weak constraints

- Weak constraints are an alternative to minimize statements
- Syntax $:\sim l_1, \dots, l_n [w@l]$
where each l_i is a literal; and w_i and p_i are integers for $1 \leq i \leq n$
- Example
 - $:\sim \text{hd}(1) . [30@2]$
 - $:\sim \text{hd}(2) . [40@2]$
 - $:\sim \text{hd}(3) . [60@2]$
 - $:\sim \text{hd}(4) . [80@2]$

Weak constraints

- Weak constraints are an alternative to minimize statements
- Syntax $: \sim l_1, \dots, l_n [w @ l]$
where each l_i is a literal; and w_i and p_i are integers for $1 \leq i \leq n$
- Example
 $: \sim \text{hd}(I, P, C) . [C @ 2]$

Outline

- 1 Motivation
- 2 Core language
- 3 Extended language
- 4 Intermediate formats**

Outline

- 1 Motivation
- 2 Core language
- 3 Extended language
- 4 Intermediate formats
 - smodels format
 - aspif format

smodels format

- The *smodels* format consists of
 - normal rules
 - choice rules
 - cardinality rules
 - weight rules
 - minimization statements
- Block-oriented format
- Note Minimization statements are not part of the logic program

smodels format

- The *smodels* format consists of
 - normal rules
 - choice rules
 - cardinality rules
 - weight rules
 - minimization statements
- Block-oriented format
- Note Minimization statements are not part of the logic program

smodels format in detail

Type/Format
Normal rule Slide ?? $1 \iota(a_0) \iota n \iota n - m \iota \iota \iota(a_{m+1}) \iota \dots \iota \iota(a_n) \iota \iota(a_1) \iota \dots \iota \iota(a_m)$
Cardinality rule Slide 21 $2 \iota(a_0) \iota n \iota n - m \iota \iota \iota(a_{m+1}) \iota \dots \iota \iota(a_n) \iota \iota(a_1) \iota \dots \iota \iota(a_m)$
Choice rule Slide 13 $3 m \iota(a_1) \iota \dots \iota \iota(a_m) \iota o - m \iota o - n \iota \iota(a_{n+1}) \iota \dots \iota \iota(a_o) \iota \iota(a_{m+1}) \iota \dots \iota \iota(a_n)$
Weight rule Slide 51 $5 \iota(a_0) \iota \iota n \iota n - m \iota \iota(a_{m+1}) \iota \dots \iota \iota(a_n) \iota \iota(a_1) \iota \dots \iota \iota(a_m) \iota w_{m+1} \iota \dots \iota w_n \iota w_1 \iota \dots \iota w_m$
Minimize statement ¹ Slide 67 $6 \iota o \iota n \iota n - m \iota \iota(a_{m+1}) \iota \dots \iota \iota(a_n) \iota \iota(a_1) \iota \dots \iota \iota(a_m) \iota w_{m+1} \iota \dots \iota w_n \iota w_1 \iota \dots \iota w_m$
Disjunctive rule Slide ?? $8 m \iota(a_1) \iota \dots \iota \iota(a_m) \iota o - m \iota o - n \iota \iota(a_{n+1}) \iota \dots \iota \iota(a_o) \iota \iota(a_{m+1}) \iota \dots \iota \iota(a_n)$

- The function ι represents a mapping of atoms to numbers

Outline

- 1 Motivation
- 2 Core language
- 3 Extended language
- 4 Intermediate formats
 - **smodels** format
 - **aspif** format

aspif format

- The *aspif* format consists of
 - rule statements
 - minimize statements
 - projection statements
 - output statements
 - external statements
 - assumption statements
 - heuristic statements
 - edge statements
 - theory terms and atoms
 - comments
- Line-oriented format

Rule statements

Rule statements have the form

$$1 _ H _ B$$

- Head H has form

$$h _ m _ a_1 _ \dots _ a_m$$

- $h \in \{0, 1\}$ determines whether the head is a disjunction or choice,
- $m \geq 0$ is the number of head elements, and
- each a_i is a positive literal

Heads are disjunctions or choices, including the special case of singular disjunctions for representing normal rules.

- Body B has one of two forms

- normal bodies have form

$$0 _ n _ l_1 _ \dots _ l_n$$

- $n \geq 0$ is the length of the rule body, and
- each l_i is a literal.

- weight bodies have form

$$1 _ l _ n _ l_1 _ w_1 _ \dots _ l_n _ w_n$$

- l is a positive integer to denote the lower bound,
- $n \geq 0$ is the number of literals in the rule body, and
- each l_i and w_i are a literal and a positive integer

Rule statements

Rule statements have the form

$$1 \sqcup H \sqcup B$$

■ Head H has form

$$h \sqcup m \sqcup a_1 \sqcup \dots \sqcup a_m$$

- $h \in \{0, 1\}$ determines whether the head is a disjunction or choice,
- $m \geq 0$ is the number of head elements, and
- each a_i is a positive literal

Heads are disjunctions or choices, including the special case of singular disjunctions for representing normal rules.

■ Body B has one of two forms

■ normal bodies have form

$$0 \sqcup n \sqcup l_1 \sqcup \dots \sqcup l_n$$

- $n \geq 0$ is the length of the rule body, and
- each l_i is a literal.

■ weight bodies have form

$$1 \sqcup l \sqcup n \sqcup l_1 \sqcup w_1 \sqcup \dots \sqcup l_n \sqcup w_n$$

- l is a positive integer to denote the lower bound,
- $n \geq 0$ is the number of literals in the rule body, and
- each l_i and w_i are a literal and a positive integer

Rule statements

Rule statements have the form

$$1 \sqcup H \sqcup B$$

■ Head H has form

$$h \sqcup m \sqcup a_1 \sqcup \dots \sqcup a_m$$

- $h \in \{0, 1\}$ determines whether the head is a disjunction or choice,
- $m \geq 0$ is the number of head elements, and
- each a_i is a positive literal

Heads are disjunctions or choices, including the special case of singular disjunctions for representing normal rules.

■ Body B has one of two forms

■ normal bodies have form

$$0 \sqcup n \sqcup l_1 \sqcup \dots \sqcup l_n$$

- $n \geq 0$ is the length of the rule body, and
- each l_i is a literal.

■ weight bodies have form

$$1 \sqcup l \sqcup n \sqcup l_1 \sqcup w_1 \sqcup \dots \sqcup l_n \sqcup w_n$$

- l is a positive integer to denote the lower bound,
- $n \geq 0$ is the number of literals in the rule body, and
- each l_i and w_i are a literal and a positive integer

Example

```
{a}.
```

```
b :- a.
```

```
c :- not a.
```

```
asp 1 0 0
```

```
1 1 1 1 0 0
```

```
1 0 1 2 0 1 1
```

```
1 0 1 3 0 1 -1
```

```
4 1 a 1 1
```

```
4 1 b 1 2
```

```
4 1 c 1 3
```

```
0
```

Example

```
{a}.
```

```
b :- a.
```

```
c :- not a.
```

```
asp 1 0 0
```

```
1 1 1 1 0 0
```

```
1 0 1 2 0 1 1
```


```
1 0 1 3 0 1 -1
```

```
4 1 a 1 1
```

```
4 1 b 1 2
```

```
4 1 c 1 3
```

```
0
```


- [1] Y. Babovich and V. Lifschitz.
Computing answer sets using program completion.
Unpublished draft, 2003.
- [2] C. Baral.
Knowledge Representation, Reasoning and Declarative Problem Solving.
Cambridge University Press, 2003.
- [3] C. Baral, G. Brewka, and J. Schlipf, editors.
Proceedings of the Ninth International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR'07), volume 4483 of *Lecture Notes in Artificial Intelligence*. Springer-Verlag, 2007.
- [4] C. Baral and M. Gelfond.
Logic programming and knowledge representation.
Journal of Logic Programming, 12:1–80, 1994.
- [5] S. Baselice, P. Bonatti, and M. Gelfond.
Towards an integration of answer set and constraint solving  Potassco

In M. Gabbrielli and G. Gupta, editors, *Proceedings of the Twenty-first International Conference on Logic Programming (ICLP'05)*, volume 3668 of *Lecture Notes in Computer Science*, pages 52–66. Springer-Verlag, 2005.

[6] A. Biere.

Adaptive restart strategies for conflict driven SAT solvers.

In H. Kleine Büning and X. Zhao, editors, *Proceedings of the Eleventh International Conference on Theory and Applications of Satisfiability Testing (SAT'08)*, volume 4996 of *Lecture Notes in Computer Science*, pages 28–33. Springer-Verlag, 2008.

[7] A. Biere.

PicoSAT essentials.

Journal on Satisfiability, Boolean Modeling and Computation, 4:75–97, 2008.

[8] A. Biere, M. Heule, H. van Maaren, and T. Walsh, editors.

Handbook of Satisfiability, volume 185 of *Frontiers in Artificial Intelligence and Applications*.

IOS Press, 2009.

- [9] G. Brewka, T. Eiter, and M. Truszczynski.
Answer set programming at a glance.
Communications of the ACM, 54(12):92–103, 2011.
- [10] G. Brewka, I. Niemelä, and M. Truszczynski.
Answer set optimization.
In G. Gottlob and T. Walsh, editors, *Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence (IJCAI'03)*, pages 867–872. Morgan Kaufmann Publishers, 2003.
- [11] K. Clark.
Negation as failure.
In H. Gallaire and J. Minker, editors, *Logic and Data Bases*, pages 293–322. Plenum Press, 1978.
- [12] M. D'Agostino, D. Gabbay, R. Hähnle, and J. Posegga, editors.
Handbook of Tableau Methods.
Kluwer Academic Publishers, 1999.

- [13] E. Dantsin, T. Eiter, G. Gottlob, and A. Voronkov.
Complexity and expressive power of logic programming.
In *Proceedings of the Twelfth Annual IEEE Conference on Computational Complexity (CCC'97)*, pages 82–101. IEEE Computer Society Press, 1997.
- [14] M. Davis, G. Logemann, and D. Loveland.
A machine program for theorem-proving.
Communications of the ACM, 5:394–397, 1962.
- [15] M. Davis and H. Putnam.
A computing procedure for quantification theory.
Journal of the ACM, 7:201–215, 1960.
- [16] E. Di Rosa, E. Giunchiglia, and M. Maratea.
Solving satisfiability problems with preferences.
Constraints, 15(4):485–515, 2010.
- [17] C. Drescher, M. Gebser, T. Grote, B. Kaufmann, A. König, M. Ostrowski, and T. Schaub.

Conflict-driven disjunctive answer set solving.

In G. Brewka and J. Lang, editors, *Proceedings of the Eleventh International Conference on Principles of Knowledge Representation and Reasoning (KR'08)*, pages 422–432. AAAI Press, 2008.

- [18] C. Drescher, M. Gebser, B. Kaufmann, and T. Schaub.

Heuristics in conflict resolution.

In M. Pagnucco and M. Thielscher, editors, *Proceedings of the Twelfth International Workshop on Nonmonotonic Reasoning (NMR'08)*, number UNSW-CSE-TR-0819 in School of Computer Science and Engineering, The University of New South Wales, Technical Report Series, pages 141–149, 2008.

- [19] N. Eén and N. Sörensson.

An extensible SAT-solver.

In E. Giunchiglia and A. Tacchella, editors, *Proceedings of the Sixth International Conference on Theory and Applications of Satisfiability Testing (SAT'03)*, volume 2919 of *Lecture Notes in Computer Science*, pages 502–518. Springer-Verlag, 2004.

- [20] T. Eiter and G. Gottlob.
On the computational cost of disjunctive logic programming:
Propositional case.
Annals of Mathematics and Artificial Intelligence, 15(3-4):289–323,
1995.
- [21] T. Eiter, G. Ianni, and T. Krennwallner.
Answer Set Programming: A Primer.
In S. Tessaris, E. Franconi, T. Eiter, C. Gutierrez, S. Handschuh,
M. Rousset, and R. Schmidt, editors, *Fifth International Reasoning
Web Summer School (RW'09)*, volume 5689 of *Lecture Notes in
Computer Science*, pages 40–110. Springer-Verlag, 2009.
- [22] F. Fages.
Consistency of Clark's completion and the existence of stable models.
Journal of Methods of Logic in Computer Science, 1:51–60, 1994.
- [23] P. Ferraris.
Answer sets for propositional theories.

In C. Baral, G. Greco, N. Leone, and G. Terracina, editors, *Proceedings of the Eighth International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR'05)*, volume 3662 of *Lecture Notes in Artificial Intelligence*, pages 119–131. Springer-Verlag, 2005.

[24] P. Ferraris and V. Lifschitz.

Mathematical foundations of answer set programming.

In S. Artëmov, H. Barringer, A. d'Avila Garcez, L. Lamb, and J. Woods, editors, *We Will Show Them! Essays in Honour of Dov Gabbay*, volume 1, pages 615–664. College Publications, 2005.

[25] M. Fitting.

A Kripke-Kleene semantics for logic programs.

Journal of Logic Programming, 2(4):295–312, 1985.

[26] M. Gebser, A. Harrison, R. Kaminski, V. Lifschitz, and T. Schaub.

Abstract Gringo.

Theory and Practice of Logic Programming, 15(4-5):449–463, 2015.

Available at <http://arxiv.org/abs/1507.06576>.

- [27] M. Gebser, R. Kaminski, B. Kaufmann, M. Lindauer, M. Ostrowski, J. Romero, T. Schaub, and S. Thiele.
Potassco User Guide.
University of Potsdam, second edition edition, 2015.
- [28] M. Gebser, R. Kaminski, B. Kaufmann, M. Ostrowski, T. Schaub, and S. Thiele.
A user's guide to gringo, clasp, clingo, and iclingo.
- [29] M. Gebser, R. Kaminski, B. Kaufmann, M. Ostrowski, T. Schaub, and S. Thiele.
Engineering an incremental ASP solver.
In M. Garcia de la Banda and E. Pontelli, editors, *Proceedings of the Twenty-fourth International Conference on Logic Programming (ICLP'08)*, volume 5366 of *Lecture Notes in Computer Science*, pages 190–205. Springer-Verlag, 2008.
- [30] M. Gebser, R. Kaminski, B. Kaufmann, and T. Schaub.

On the implementation of weight constraint rules in conflict-driven ASP solvers.

In Hill and Warren [49], pages 250–264.

[31] M. Gebser, R. Kaminski, B. Kaufmann, and T. Schaub.

Answer Set Solving in Practice.

Synthesis Lectures on Artificial Intelligence and Machine Learning.
Morgan and Claypool Publishers, 2012.

[32] M. Gebser, B. Kaufmann, A. Neumann, and T. Schaub.

clasp: A conflict-driven answer set solver.

In Baral et al. [3], pages 260–265.

[33] M. Gebser, B. Kaufmann, A. Neumann, and T. Schaub.

Conflict-driven answer set enumeration.

In Baral et al. [3], pages 136–148.

[34] M. Gebser, B. Kaufmann, A. Neumann, and T. Schaub.

Conflict-driven answer set solving.

In Veloso [74], pages 386–392.

- [35] M. Gebser, B. Kaufmann, A. Neumann, and T. Schaub.
Advanced preprocessing for answer set solving.
In M. Ghallab, C. Spyropoulos, N. Fakotakis, and N. Avouris, editors, *Proceedings of the Eighteenth European Conference on Artificial Intelligence (ECAI'08)*, pages 15–19. IOS Press, 2008.
- [36] M. Gebser, B. Kaufmann, and T. Schaub.
The conflict-driven answer set solver clasp: Progress report.
In E. Erdem, F. Lin, and T. Schaub, editors, *Proceedings of the Tenth International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR'09)*, volume 5753 of *Lecture Notes in Artificial Intelligence*, pages 509–514. Springer-Verlag, 2009.
- [37] M. Gebser, B. Kaufmann, and T. Schaub.
Solution enumeration for projected Boolean search problems.
In W. van Hoes and J. Hooker, editors, *Proceedings of the Sixth International Conference on Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems*

(CPAIOR'09), volume 5547 of *Lecture Notes in Computer Science*, pages 71–86. Springer-Verlag, 2009.

[38] M. Gebser, M. Ostrowski, and T. Schaub.

Constraint answer set solving.

In Hill and Warren [49], pages 235–249.

[39] M. Gebser and T. Schaub.

Tableau calculi for answer set programming.

In S. Etalle and M. Truszczynski, editors, *Proceedings of the Twenty-second International Conference on Logic Programming (ICLP'06)*, volume 4079 of *Lecture Notes in Computer Science*, pages 11–25. Springer-Verlag, 2006.

[40] M. Gebser and T. Schaub.

Generic tableaux for answer set programming.

In V. Dahl and I. Niemelä, editors, *Proceedings of the Twenty-third International Conference on Logic Programming (ICLP'07)*, volume 4670 of *Lecture Notes in Computer Science*, pages 119–133.

Springer-Verlag, 2007.

- [41] M. Gelfond.
Answer sets.
In V. Lifschitz, F. van Harmelen, and B. Porter, editors, *Handbook of Knowledge Representation*, chapter 7, pages 285–316. Elsevier Science, 2008.
- [42] M. Gelfond and Y. Kahl.
Knowledge Representation, Reasoning, and the Design of Intelligent Agents: The Answer-Set Programming Approach.
Cambridge University Press, 2014.
- [43] M. Gelfond and N. Leone.
Logic programming and knowledge representation — the A-Prolog perspective.
Artificial Intelligence, 138(1-2):3–38, 2002.
- [44] M. Gelfond and V. Lifschitz.
The stable model semantics for logic programming.

In R. Kowalski and K. Bowen, editors, *Proceedings of the Fifth International Conference and Symposium of Logic Programming (ICLP'88)*, pages 1070–1080. MIT Press, 1988.

[45] M. Gelfond and V. Lifschitz.

Logic programs with classical negation.

In D. Warren and P. Szeredi, editors, *Proceedings of the Seventh International Conference on Logic Programming (ICLP'90)*, pages 579–597. MIT Press, 1990.

[46] E. Giunchiglia, Y. Lierler, and M. Maratea.

Answer set programming based on propositional satisfiability.

Journal of Automated Reasoning, 36(4):345–377, 2006.

[47] K. Gödel.

Zum intuitionistischen Aussagenkalkül.

In *Anzeiger der Akademie der Wissenschaften in Wien*, page 65–66. 1932.

[48] A. Heyting.

Die formalen Regeln der intuitionistischen Logik.

In *Sitzungsberichte der Preussischen Akademie der Wissenschaften*, page 42–56. 1930.

Reprint in *Logik-Texte: Kommentierte Auswahl zur Geschichte der Modernen Logik*, Akademie-Verlag, 1986.

[49] P. Hill and D. Warren, editors.

Proceedings of the Twenty-fifth International Conference on Logic Programming (ICLP'09), volume 5649 of *Lecture Notes in Computer Science*. Springer-Verlag, 2009.

[50] J. Huang.

The effect of restarts on the efficiency of clause learning.

In Veloso [74], pages 2318–2323.

[51] K. Konczak, T. Linke, and T. Schaub.

Graphs and colorings for answer set programming.

Theory and Practice of Logic Programming, 6(1-2):61–106, 2006.

[52] J. Lee.

A model-theoretic counterpart of loop formulas.

In L. Kaelbling and A. Saffiotti, editors, *Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence (IJCAI'05)*, pages 503–508. Professional Book Center, 2005.

- [53] N. Leone, G. Pfeifer, W. Faber, T. Eiter, G. Gottlob, S. Perri, and F. Scarcello.

The DLV system for knowledge representation and reasoning.

ACM Transactions on Computational Logic, 7(3):499–562, 2006.

- [54] V. Lifschitz.

Answer set programming and plan generation.

Artificial Intelligence, 138(1-2):39–54, 2002.

- [55] V. Lifschitz.

Introduction to answer set programming.

Unpublished draft, 2004.

- [56] V. Lifschitz and A. Razborov.

Why are there so many loop formulas?

ACM Transactions on Computational Logic, 7(2):261–268, 2006.

- [57] F. Lin and Y. Zhao.
ASSAT: computing answer sets of a logic program by SAT solvers.
Artificial Intelligence, 157(1-2):115–137, 2004.
- [58] V. Marek and M. Truszczyński.
Nonmonotonic logic: context-dependent reasoning.
Artificial Intelligence. Springer-Verlag, 1993.
- [59] V. Marek and M. Truszczyński.
Stable models and an alternative logic programming paradigm.
In K. Apt, V. Marek, M. Truszczyński, and D. Warren, editors, *The Logic Programming Paradigm: a 25-Year Perspective*, pages 375–398. Springer-Verlag, 1999.
- [60] J. Marques-Silva, I. Lynce, and S. Malik.
Conflict-driven clause learning SAT solvers.
In Biere et al. [8], chapter 4, pages 131–153.
- [61] J. Marques-Silva and K. Sakallah.

GRASP: A search algorithm for propositional satisfiability.

IEEE Transactions on Computers, 48(5):506–521, 1999.

[62] V. Mellarkod and M. Gelfond.

Integrating answer set reasoning with constraint solving techniques.

In J. Garrigue and M. Hermenegildo, editors, *Proceedings of the Ninth International Symposium on Functional and Logic Programming (FLOPS'08)*, volume 4989 of *Lecture Notes in Computer Science*, pages 15–31. Springer-Verlag, 2008.

[63] V. Mellarkod, M. Gelfond, and Y. Zhang.

Integrating answer set programming and constraint logic programming.

Annals of Mathematics and Artificial Intelligence, 53(1-4):251–287, 2008.

[64] D. Mitchell.

A SAT solver primer.

Bulletin of the European Association for Theoretical Computer Science, 85:112–133, 2005.

- [65] M. Moskewicz, C. Madigan, Y. Zhao, L. Zhang, and S. Malik.
Chaff: Engineering an efficient SAT solver.
In *Proceedings of the Thirty-eighth Conference on Design Automation (DAC'01)*, pages 530–535. ACM Press, 2001.
- [66] I. Niemelä.
Logic programs with stable model semantics as a constraint programming paradigm.
Annals of Mathematics and Artificial Intelligence, 25(3-4):241–273, 1999.
- [67] R. Nieuwenhuis, A. Oliveras, and C. Tinelli.
Solving SAT and SAT modulo theories: From an abstract Davis-Putnam-Logemann-Loveland procedure to DPLL(T).
Journal of the ACM, 53(6):937–977, 2006.
- [68] K. Pipatsrisawat and A. Darwiche.
A lightweight component caching scheme for satisfiability solvers.

In J. Marques-Silva and K. Sakallah, editors, *Proceedings of the Tenth International Conference on Theory and Applications of Satisfiability Testing (SAT'07)*, volume 4501 of *Lecture Notes in Computer Science*, pages 294–299. Springer-Verlag, 2007.

- [69] L. Ryan.
Efficient algorithms for clause-learning SAT solvers.
Master's thesis, Simon Fraser University, 2004.
- [70] P. Simons, I. Niemelä, and T. Soinen.
Extending and implementing the stable model semantics.
Artificial Intelligence, 138(1-2):181–234, 2002.
- [71] T. Son and E. Pontelli.
Planning with preferences using logic programming.
Theory and Practice of Logic Programming, 6(5):559–608, 2006.
- [72] T. Syrjänen.
Lparse 1.0 user's manual, 2001.
- [73] A. Van Gelder, K. Ross, and J. Schlipf.

The well-founded semantics for general logic programs.

Journal of the ACM, 38(3):620–650, 1991.

[74] M. Veloso, editor.

Proceedings of the Twentieth International Joint Conference on Artificial Intelligence (IJCAI'07). AAAI/MIT Press, 2007.

[75] L. Zhang, C. Madigan, M. Moskewicz, and S. Malik.

Efficient conflict driven learning in a Boolean satisfiability solver.

In *Proceedings of the International Conference on Computer-Aided Design (ICCAD'01)*, pages 279–285. ACM Press, 2001.