Answer Set Programming

(2) Extensions of ASP programs

Abdallah Saffidine

COMP4418

Overview of the Lecture

- Semantics of ASP programs
- **Extensions of ASP programs**
- Handling of variables in ASP
- ASP as modelling language

Choice Rules

Definition: choice rule

A **choice rule** is a rule the form

$$\{A_1; \ldots; A_k\} \leftarrow B_1,\ \ldots, B_m, \mathrm{not}\, C_1,\ \ldots, \mathrm{not}\, C_n$$

which allows any subset of $\{A_1, \ldots, A_k\}$ in a stable model.

Choice Rules

Definition: choice rule

A **choice rule** is a rule the form

$$\{A_1; \ldots; A_k\} \leftarrow B_1,\ \ldots, B_m, \mathrm{not}\, C_1,\ \ldots, \mathrm{not}\, C_n$$

which allows any subset of $\{A_1, \ldots, A_k\}$ in a stable model.

Theorem: reduction to normal rules

A choice rule can be encoded by $2k+1$ normal rules using $2k+1$ new atoms.

Choice Rules

Definition: choice rule

A **choice rule** is a rule the form

$$\{A_1; \dots; A_k\} \leftarrow B_1, \dots, B_m, \text{not } C_1, \dots, \text{not } C_n$$

which allows any subset of $\{A_1, \dots, A_k\}$ in a stable model.

Theorem: reduction to normal rules

A choice rule can be encoded by $2k + 1$ normal rules using $2k + 1$ new atoms.

Further extensions:

- Conditional literals: $\{A : B\}$
  Ex.: $\{m(v, C) : c(C)\}$ expands to $\{m(v, r); m(v, g); m(v, b)\}$
- Cardinality constraints: $min\ \{A_1; \dots; A_k\}\ max$
  Ex.: $1\ \{m(v, r); m(v, g); m(v, b)\}\ 1$

Integrity Constraints

Definition: integrity constraint

An **integrity constraint** is a rule r of the form

$$\leftarrow B_1, \ldots, B_m, \mathrm{not}\, C_1, \ldots, \mathrm{not}\, C_n$$

S **satisfies** r iff some $B_i \notin S$ or some $C_j \in S$.

P^S contains $\leftarrow B_1, \ldots, B_m$ iff P contains r and $C_1, \ldots, C_n \notin S$.

Integrity Constraints

Definition: integrity constraint

An **integrity constraint** is a rule r of the form

$$\leftarrow B_1, \ldots, B_m, \text{not}\, C_1, \ldots, \text{not}\, C_n$$

S **satisfies** r iff some $B_i \notin S$ or some $C_j \in S$.

P^S contains $\leftarrow B_1, \ldots, B_m$ iff P contains r and $C_1, \ldots, C_n \notin S$.

Theorem: reduction to normal rules

Let P' be like P except that every integrity constraint

$$\leftarrow B_1, \ldots, B_m, \text{not}\, C_1, \ldots, \text{not}\, C_n$$

is replaced with

$$\mathit{dummy} \leftarrow B_1, \ldots, B_m, \text{not}\, C_1, \ldots, \text{not}\, C_n, \text{not}\, \mathit{dummy}$$

for some new atom dummy.

Then P and P' have the same stable models.

Negation in the Rule Head

Definition: rules with negated head

A rule with **negated head** is of the form

$$\text{not}\, A \leftarrow B_1, \ldots, B_m, \text{not}\, C_1, \ldots, \text{not}\, C_n$$

Negation in the Rule Head

Definition: rules with negated head

A rule with **negated head** is of the form

$$\text{not}\, A \leftarrow B_1, \ldots, B_m, \text{not}\, C_1, \ldots, \text{not}\, C_n$$

Theorem: reduction to normal rules

Let P' be like P except that every rule with negated head

$$\text{not}\, A \leftarrow B_1, \ldots, B_m, \text{not}\, C_1, \ldots, \text{not}\, C_n$$

is replaced with

$$\leftarrow B_1, \ldots, B_m, \text{not}\, C_1, \ldots, \text{not}\, C_n, \text{not}\, \mathit{dummy}$$

and

$$\mathit{dummy} \leftarrow \text{not}\, A$$

for some new atom dummy.

Then P and P' have the same stable models (modulo dummy propositions).

Complexity

Theorem: complexity of NLPs without negations

Is S a stable model of a negation-free P? – **Linear time**
Does a negation-free P have a stable model? – **Constant** (yes, one)

Theorem: complexity of NLPs with negations

Is S a stable model of P? – **Linear time**
Does P have a stable model? – **NP-complete**

Note: integrity constraints, choice rules, negation in heads **preserve complexity** (program grows only polynomially)