Mathematical Foundations of Answer Set Programming

Authors: Paolo Ferraris and Vladimir Lifschitz

Presenter: Yana Maximova Todorova

September 26^{th}, 2008

Talk Outline

- **Introduction**
- **Stable Models**
- **Programming**
- **Proofs of Theorems**
- **Conclusion**
- **Appendix A: Propositional Logic**
- **Appendix B: Traditional Definition of a Stable Model**

Introduction of ASP

- Form of declarative logic programming
- Oriented towards difficult combinatorial search problems
- Examples of applications:
  1) developing a decision support system for the Space Shuttle
  2) graph-theoretic problems arising in zoology and linguistics
- Syntax of ASP programs: similar to Prolog programs
- Computational mechanisms used in ASP: based on ideas that have led to the creation of satisfiability solvers for propositional logic

History of ASP

- Emerged from interaction between research on:
  1) semantics of negation in logic programming
  2) applications of satisfiability solvers to search problems

- Identified as a new programming paradigm in 1999

- Main definition of ASP: tells under what conditions a model of a propositional formula F is called $stable$

- Idea of ASP:
  1) represent the search problem of interest as the problem of finding a stable model of a formula
  2) find a solution using an $answer\ set\ solver$ (SMODELS, DLV, etc.)

Stable Models: Motivation

- Two equivalent formulas do not necessarily have the same stable models:
  (1) $\neg p \rightarrow q$ has $\{q\}$ as the only stable model
  (2) $\neg q \rightarrow p$ has $\{p\}$ as the only stable model

- Formulas (1) and (2) are equivalent classically, but not intuitionistically

- Intuitionistically equivalent formulas always have the same stable models

Stable models (ctd)

- A conjunction $F \wedge G$ may have a stable model that is not a stable model of F:
  (3) $(\neg p \rightarrow q) \wedge p$ has one stable model $\{p\}$, which is $\neq \{q\}$

- Appending an additional conjunctive term to a formula may give it a new stable model

- The concept of a stable model is nonmonotonic

- Early work on stable models was outgrowth of:
  1) Research on formal nonmonotonic reasoning
  2) Study of the relationship between autoepistemic logic and the semantics of negation in logic programming

Stable Models: Definition

- The $reduct\ F^X$ of a formula F relative to a set X of atoms is the formula obtained from F by replacing each maximal subformula that is not satisfied by X with $\bot$ [Ferraris, 2005]

- X is a $stable\ model$ (or an $answer\ set$) of F if X is minimal among the sets satisfying F^X.

- Steps to verify that X is a stable model of F:
  (i) mark in F the maximal subformulas that are not satisfied by X;
  (ii) replace each of these subformulas with $\bot$ (after that, equivalent transformations of classical propositional logic can be used to simplify the result);
  (iii) check that the resulting formula is satisfied by X;
  (iv) check that it is not satisfied by any proper subset of X.

Stable Models: Examples

Check that $\{q\}$ is an answer set of:

$$(1)\ \neg p \rightarrow q$$

(i)mark the only subformula of (1) that is not satisfied by $\{q\}$:

$$\neg \underline{p} \rightarrow q;$$

(ii) replace that subformula with $\bot$:

$$\neg \bot \rightarrow q;$$

simplify:

$$q;$$

(iii) check that the last formula is satisfied by $\{q\}$;

(iv) check that it is not satisfied by $\emptyset$.

Theorem 1

- An occurrence of an atom A in a formula F is $strictly\ positive$ if it does not belong to the antecedent of any implication in F
- An atom A is a $head\ atom$ of a formula F if at least one occurrence of A in F is strictly positive
- THEOREM 1 ([Ferraris, 2005]). $Any\ stable\ model\ of\ F\ is\ a\ subset\ of\ the\ set\ of\ head\ atoms\ of\ F$.

Recursive definition of the reduct

For any X

$$\bot^X = \bot;$$

$$A^X = \begin{cases} A, & \text{if X} \models A, \\ \bot, & \text{otherwise} \end{cases}$$

(A is an atom);

$$(F \odot G)^X = \begin{cases} F^X \odot G^X, & \text{if X} \models F \odot G, \\ \bot, & \text{otherwise} \end{cases}$$

($\odot$ is $\wedge$, $\vee$ or $\rightarrow$).

$$(\neg F)^X = \begin{cases} \bot, & \text{if X} \models F, \\ \top, & \text{otherwise.} \end{cases}$$

Horn Formulas

- A $Horn\ formula$ is a conjunction of several (0 or more) implications of the form $F \rightarrow A$, where F is a conjunction of several (0 or more) atoms, and A is an atom.

- For any Horn formula F, the intersection of all models of F is a model of F also; it is called the $minimal\ model$ of F.

- THEOREM 2. $For\ any\ Horn\ formula\ F, the\ minimal\ model\ of\ F\ is\ the\ only\ stable\ model\ of\ F$.

Horn Formulas: Examples

- The formula

  (5) $p \wedge (p \rightarrow q) \wedge (q \wedge r \rightarrow s)$

  has one stable model: its minimal model $\{p, q\}$.

- The only model of the empty conjunction $\top$ is the empty set.

Choice Formulas

For any finite set Z of atoms, by Z^c we denote the formula

$$\bigwedge_{A \in Z} (A \vee \neg A).$$

- PROPOSITION 3. *For any finite set Z of atoms, a set X of atoms is a stable model of Z^c iff $X \subseteq Z$.*

Proof of Proposition 3

- For any subset X of Z, the reduct of Z^c relative to X is

$$\bigwedge_{A \in X} (A \vee \bot) \wedge \bigwedge_{A \in Z \setminus X} (\bot \vee \neg \bot),$$

- which is equivalent to $\bigwedge_{A \in X} A$.

- This formula is satisfied by X, but is not satisfied by any proper subset of X.

- Therefore, if X is a subset of Z then X is a stable model of Z^c.

- The converse is immediate from Theorem 1.

Choice formula: Example

- Choice formula $\{p, q\}^c$ is
$$(p \vee \neg p) \wedge (q \vee \neg q).$$
- It has 4 answer sets: arbitrary subsets of $\{p, q\}$.
- Generally, if Z consists of n atoms, then Z^c has 2^n stable models.
- To form a model, we choose for every element of Z arbitrarily whether to include it in the model.

Constraints

- Art of ASP: based on possibility of representing the collection of sets that we are interested in as the collection of stable models of a formula

- Conjoin a choice formula (which provides approximation from above for the collection of sets that we want to describe) with constraints (that eliminate the unsuitable stable models)

- A $constraint$ is simply a formula beginning with negation.

- PROPOSITION 4. $A\ set\ of\ atoms\ is\ a\ stable\ model\ of\ F \wedge \neg G\ iff\ it\ is\ a\ stable\ model\ of\ F\ that\ satisfies\ \neg G$.

Proof of Proposition 4

- Case 1: X satisfies $F \wedge \neg G$.
  Then X does not satisfy G, and
  $(F \wedge \neg G)^X$ is $F^X \wedge \neg \bot$,
  which is equivalent to F^X.
  Therefore, X is minimal among the sets satisfying F^X iff it is minimal among the sets satisfying $(F \wedge \neg G)^X$.

- Case 2: X does not satisfy $F \wedge \neg G$.
  Then X cannot be a model of F that satisfies $\neg G$.

Constraint: Example

- Recall that the stable model of
  (1) $\neg p \rightarrow q$ is $\{q\}$.
- If we conjoin (1) with the constraint $\neg p$, we get formula
  (2) $(\neg p \rightarrow q) \wedge \neg p$.
- Since the only stable model $\{q\}$ of (1) satisfies the constraint, the conjunction (2) has $\{q\}$ as the only stable model as well.
- If we conjoin (1) with the constraint $\neg\neg p$, then we get formula
  (3) $(\neg p \rightarrow q) \wedge \neg\neg p$.
- Since the only stable model $\{q\}$ of (1) does not satisfy this constraint, conjunction (3) has no stable models.

LPARSE and SMODELS

- SMODELS is one of the widely used answer set solvers.
- Its frontend LPARSE serves also as the frontend of 3 other systems for computing stable models: GnT, ASSAT, and CMODELS.
- LPARSE requires that the input formula be represented in a special format, as a conjunction of ”rules”, similar to Prolog rules.

Representing Horn formulas in LPARSE

- Formula
  (1) $p \wedge (p \rightarrow q) \wedge (q \wedge r \rightarrow s)$
  would be represented in LPARSE $input$ file as:

```
p.
q :- p.
s :- q, r.
```

- To instruct SMODELS to find stable model of (1), we invoke LPARSE and SMODELS as:

```
% lparse input | smodels
```

- The corresponding output is:

```
Answer: 1
Stable Model: q p
```

Representing negation in LPARSE

- Formula
  (1) $(\neg p \rightarrow q) \wedge (\neg q \rightarrow p)$
  would be written in file $input$ as:

```
q :- not p.
p :- not q.
```

- The command line that instructs SMODELS to find the stable models is:

```
% lparse input | smodels 0
```

- The produced output is:

```
Answer: 1
Stable Model: q
Answer: 2
Stable Model: p
```

Representing choice formulas and constraints in LPARSE

- To represent a choice formula $\{A_1, \ldots, A_n\}^c$ in LPARSE, we simply drop the superscript ^{c}.

- A constraint $\neg F$, where F is a conjunction of literals, is written as
  `:- F.`

- Formula $\{p, q, r\}^c \wedge \neg\neg p \wedge \neg(q \wedge \neg r)$ can be written as
  ```
  {p,q,r}.
  :- not p.
  :- q, not r.
  ```

Search process of SMODELS

- Search process is sophisticated; it guarantees that every stable model of the given input will be found.
- It can be viewed as a modification of the Davis-Putman-Logemann-Loveland procedure for the propositional satisfiability problem (SAT).
- Finding a stable model of a formula is more difficult than SAT: the existence of a stable model is a Σ_2^P-complete property.
- However, most uses of ASP involve formulas of special syntactic forms for which this property is known to be in class NP.
- Systems ASSAT and CMODELS reduce the problem of computing stable models of a given formula to an instance of SAT and then invoke SAT solvers to do the search.

Strong Equivalence

- A formula F is $strongly\ equivalent$ to a formula G if any formula $F^{'}$ that contains an occurrence of F has the same stable models as the formula $G^{'}$ obtained from $F^{'}$ by replacing that occurrence with G.

- For example, $p \rightarrow q$ has the same stable model as $p \rightarrow r$: the empty set. But these two formulas are not strongly equivalent.

- Take $F^{'}$ to be $(p \rightarrow q) \wedge p$. Then $G^{'}$ is $(p \rightarrow r) \wedge p$. These two formulas have different stable models: $\{p, q\}$ and $\{p, r\}$ respectively.

Role of strong equivalence in ASP

- Strong equivalence allow us to simplify a part of a program without looking at the rest of it.

- For example, formula $p \wedge (p \rightarrow q)$ is intuitionistically equivalent to $p \wedge q$.

- Therefore, in any program containing the rules

  ```
  p.
  q :- p.
  ```

  replacing the second rule by

  ```
  q.
  ```

  will have no effect on the set of stable models.

Choosing a counterexample

- If formulas F and G are not strongly equivalent to each other, then this can be always demonstrated using a counterexample F' that is not much more complicated than F.

- Take F' to be a formula of the form $F \wedge H$, where H is a Horn formula.

- A formula H is $unary$ if it is a conjunction of several atoms and implications of the from $A_1 \rightarrow A_2$, where A_1 and A_2 are atoms.

- Formulas F and G are strongly equivalent iff, for every unary H, $F \wedge H$ and $G \wedge H$ have the same stable models.

Theorem 5

For any formulas F and G, the following conditions are equivalent:

(i) F is strongly equivalent to G,

(ii) for every unary formula H, $F \wedge H$ and $G \wedge H$ have the same stable models,

(iii) F is equivalent to G in the logic of here-and-there.

Examples of the use of Theorem 5

- Formulas $\neg\neg p$ and $\neg p \rightarrow p$ are intuituionistically equivalent. Therefore, the result of replacing the subformula $\neg p \rightarrow p$ in any formula with $\neg\neg p$ does not change its stable models. That is, any program containing the rule

  ```
  p :- not p.
  ```

  can be simplified by replacing that rule with

  ```
  :- not p.
  ```

- Formulas $p \vee \neg p$ and $\neg\neg p \rightarrow p$ are equivalent to each other in the logic of here-and-there. Therefore, they are strongly equivalent. The first one can be written in LPARSE as

  ```
  {p}.
  ```

  The second formula can be written as

  ```
  p :- {not p} 0.
  ```

Another use of Theorem 5

- PROPOSITION 6. *Let Z be the set of atoms occurring in a formula F. A subset X of Z satisfies F iff X is a stable model of $Z^c \wedge F$.*

- Proof.
  From Propositions 3 and 4, a subset X of Z satisfies F iff X is a stable model of $Z^c \wedge \neg\neg F$. It remains to observe that $\neg\neg F \leftrightarrow F$ can be intuitionistically derived from Z^c, because Z^c is the conjunction of the excluded middle formulas $A \vee \neg A$ for all atoms A occurring in this equivalence.

- Proposition 6 provides a reduction of the propositional satisfiability problem to ASP: to find a model of F, look for a stable model of the conjunction of F with the excluded middle formulas $A \vee \neg A$ for all atoms A occurring in F.

Theorem 7 ([Ferraris, 2005])

- *For any formulas F and G, F is strongly equivalent to G iff, for every set X of atoms, F^X is equivalent to G^X in classical logic.*

- Example: the fact that $\neg p \rightarrow p$ is strongly equivalent to $\neg\neg p$ can be established by:

$$\begin{array}{lll} (\neg p \rightarrow p)^{\{p\}} = \bot \rightarrow p & \leftrightarrow \top, \\ (\neg\neg p)^{\{p\}} \quad = \neg\bot & = \top; \end{array}$$

$$(\neg p \rightarrow p)^{\emptyset} = \bot.$$
$$(\neg\neg p)^{\emptyset} = \bot.$$

Splitting

- Consider the following example:

  (6) $\{p, q\}^c \wedge (p \rightarrow r) \wedge (q \wedge r \rightarrow s)$.

  The first conjunctive term has 4 stable models:

  (7) $\emptyset,\ \{p\},\ \{q\},\ \{p, q\}$.

- The rest of the conjunction: a "definition", characterizing r and s in terms of p and q. Appending this definition to the choice formula $\{p, q\}^c$ does not affect the total number of its stable models, but it can change each of the models (7) by adding to it some of the atoms r, s.

- Based on $p \rightarrow r$, atom r is added to each model containing p.

- Based on $q \wedge r \rightarrow s$, atom s is added to each model containing both q and r.

- Thus (6) have the following stable models:
  (8) $\emptyset,\ \{p, r\},\ \{q\},\ \{p, q, r, s\}$.

Theorem 8

- *Let F and G be formulas such that F does not contain any head atoms of G. A set X of atoms is a stable model of $F \wedge G$ iff there exists a stable model $\{A_1, \ldots, A_n\}$ of F such that X is a stable model of $A_1 \wedge \ldots \wedge A_n \wedge G$.*
- In application to $\{p, q\}^c \wedge (p \rightarrow r) \wedge (q \wedge r \rightarrow s)$, we take $\{p, q\}^c$ to be F and $(p \rightarrow r) \wedge (q \wedge r \rightarrow s)$ to be G.
- Since F does not containt any of the head atoms r, s of G, the stable models of $F \wedge G$ can be generated by taking each of the stable models of F ($\emptyset$, $\{p\}$, $\{q\}$, $\{p, q\}$), conjoining its elements with G, and listing all stable models of each of the resulting formulas

$$
\begin{aligned}
(p \rightarrow r) \wedge (q \wedge r \rightarrow s),\\
p \wedge (p \rightarrow r) \wedge (q \wedge r \rightarrow s),\\
q \wedge (p \rightarrow r) \wedge (q \wedge r \rightarrow s),\\
p \wedge q \wedge (p \rightarrow r) \wedge (q \wedge r \rightarrow s).
\end{aligned}
$$

- Since these are Horn formulas, each of them has one stable model: its minimal model.

Example of the use of Theorem 8

- Find the stable models of the conjunction

  (9) $(\neg p \rightarrow q) \wedge (q \rightarrow r)$.

- The only stable model of the first conjunctive term is $\{q\}$. According to Theorem 8, it follows that (9) has the same stable models as

  $$q \wedge (q \rightarrow r).$$

- This is a Horn formula, and its minimal model $\{q, r\}$ is its only stable model.

Proposition 9

- *For any atom A that is not a head atom of F, F has the same stable models as $F^A_\bot$.*

- Notation: F^A_G stands for the formula obtained from a formula F by substituting a formula G for all occurrences of an atom A.

- Proof.
  Since A is not a head atom of F, A does not belong to any of the stable models of F (Theorem 1).

  Therefore, F has the same stable models as $F \wedge \neg A$ (Proposition 4).

  Similarly, $F^A_\bot$ has the same stable model as $F^A_\bot \wedge \neg A$.

  It remains to observe that $F \wedge \neg A$ and $F^A_\bot \wedge \neg A$ are intuitionistically equivalent to each other by the replacement property of intuitionistic logic.

Example of the use of Proposition 9

- Find the stable model of $\neg p \longrightarrow q$.
- Since p is not a head atom of $\neg p \longrightarrow q$, this formula has the same stable models as $\neg \bot \longrightarrow q$, which is intuitionistically equivalent to the Horn formula q.
- Consequently, the only stable model of $\neg p \longrightarrow q$ is q.

Proposition 10

- *For any atom A, a set X of atoms is a stable model of $F \wedge A$ iff there exists a stable model Y of $F^A_\top$ such that $X = Y \cup \{A\}$.*

- Proof.

  By the replacement property of intuitionistic logic, $F \wedge A$ is intuitionistically equivalent to $F^A_\top$, so that the two formulas have the same stable models.

  By Theorem 8, X is a stable model of $F^A_\top \wedge A$ iff there exists a stable model $\{A_1, \ldots, A_n\}$ of $F^A_\top$ such that X is a stable model of $A_1 \wedge \ldots \wedge A_n \wedge A$.

  The only stable model of this Horn formula is $\{A_1, \ldots, A_n, A\}$, which can be written as $\{A_1, \ldots, A_n\} \cup \{A\}$.

Example of the use of Proposition 10

- Find the stable models of $(\neg p \longrightarrow q) \wedge p$.
- We need to add p to each stable model of $\neg\top \longrightarrow q$.
- Since this formula is intuitionistically equivalent to $\top$, its only stable model is the empty set.
- Therefore, the only stable model of $(\neg p \longrightarrow q) \wedge p$ is $\{p\}$.

Example of the use of Proposition 9, 10 and Splitting

- Find the stable models of
  (1) $\{p, q\}^c \wedge (\neg p \rightarrow r)$.
- By Theorem 8, this can be done by computing the stable models of each of

$$\begin{aligned} \neg p &\rightarrow r, \\ p \wedge (\neg p &\rightarrow r), \\ q \wedge (\neg p &\rightarrow r), \\ p \wedge q \wedge (\neg p &\rightarrow r). \end{aligned}$$

- Proposition 9 shows that the only stable model of the first of these formulas is $\{r\}$.
- Proposition 10 shows that the only stable model of the second formula is $\{p\}$.
- Proposition 9 shows that the only stable model of the third formula is $\{q, r\}$.
- Proposition 10 shows that the only stable model of the last formula is $\{p, q\}$.
- Therefore, (1) has 4 stable models:

$$\{p\},\ \{r\},\ \{q, r\},\ \{p, q\}.$$

Proposition 11

- *Let F and G be formulas such that F does not contain head atoms of G, and G does not contain head atoms of F. A set of atoms is a stable model of $F \wedge G$ iff it can be represented as the union of a stable model of F and a stable model of G.*

- Proof.
  By Theorem 8, X is a stable model of $F \wedge G$ iff there exists a stable model $\{A_1, \ldots, A_n\}$ of F such that X is a stable model of

  (11) $G \wedge (A_1 \wedge \ldots \wedge A_n)$.

  By Theorem 1, for any stable model $\{A_1, \ldots, A_n\}$ of F, atoms $A_1, \ldots, A_n$ are head atoms of F. Therefore, they are different from the head atoms of G, so that the head atoms of G do not occur in the second conjunctive term of (11).

  By Theorem 8, X is a stable model of (11) iff there exists a stable model $\{B_1, \ldots, B_m\}$ of G such that X is a stable model of

  $$B_1 \wedge \ldots \wedge B_m \wedge A_1 \wedge \ldots \wedge A_n,$$

  that is to say, such that

  $$X = \{A_1, \ldots, A_n\} \cup \{B_1, \ldots, B_m\}.$$

Cardinality Expressions

- For any nonnegative integer l (lower bound) and formulas $F_1, \ldots, F_n$,

  (12) $l \leq \{F_1, \ldots, F_n\}$

  stands for the disjunction

$$\bigvee_{I \subseteq \{1,\ldots,n\},\ |I|=l} \ \bigwedge_{i \in I} F_i.$$

- For instance,

$$2 \leq \{F_1, F_2, F_3\}$$

  stands for

$$(F_1 \wedge F_2) \vee (F_1 \wedge F_3) \vee (F_2 \wedge F_3).$$

- By

  (13) $\{F_1, \ldots, F_n\} \leq u$

  where u is a nonnegative integer (upper bound) we denote the formula

$$\neg(u + 1 \leq \{F_1, \ldots, F_n\}).$$

Cardinality Expressions (ctd)

- Finally,

  (14) $l \leq \{F_1, \ldots, F_n\} \leq u$

  stands for

  $$(l \leq \{F_1, \ldots, F_n\}) \wedge (\{F_1, \ldots, F_n\} \leq u).$$

- It is clear that any set of atoms
  - satisfies (12) iff it satisfies at least l of the formulas $F_1, \ldots, F_n$;
  - satisfies (13) iff it satisfies at most u of the formulas $F_1, \ldots, F_n$;
  - satisfies (14) iff it satisfies at least l and at most u of the formulas $F_1, \ldots, F_n$.

- The input language of LPARSE allows us to use expressions (12)-(14) in the $bodies$ of rules, with the symbol $\leq$ dropped, if all formulas $F_1, \ldots, F_n$ are literals.

- For example, the implication $\neg\neg p \rightarrow p$ can be represented in LPARSE as
  `p :- {not p} 0.`

Some useful abbreviations

- If $A_1, \ldots, A_n$ are pairwise distinct atoms then we will write

$$\begin{array}{rcl} l \leq \{A_1, \ldots, A_n\}^c & \text{for} & \{A_1, \ldots, A_n\}^c \wedge (l \leq \{A_1, \ldots, A_n\}), \\ \{A_1, \ldots, A_n\}^c \leq u & \text{for} & \{A_1, \ldots, A_n\}^c \wedge (\{A_1, \ldots, A_n\} \leq u), \\ l \leq \{A_1, \ldots, A_n\}^c \leq u & \text{for} & \{A_1, \ldots, A_n\}^c \wedge (l \leq \{A_1, \ldots, A_n\} \leq u). \end{array}$$

- PROPOSITION 12. *For any pairwise distinct atoms $A_1, \ldots, A_n$, nonnegative integers l and u, and a set X of atoms,*

  (i) X is a stable model of $l \leq \{A_1, \ldots, A_n\}^c$ iff $X \subseteq \{A_1, \ldots, A_n\}$ and $l \leq |X|$;

  (ii) X is a stable model of $\{A_1, \ldots, A_n\}^c \leq u$ iff $X \subseteq \{A_1, \ldots, A_n\}$ and $|X| \leq u$;

  (iii) X is a stable model of $l \leq \{A_1, \ldots, A_n\}^c \leq u$ iff $X \subseteq \{A_1, \ldots, A_n\}$ and $l \leq |X| \leq u$.

- Proof: Immediate from Proposition 6.

Examples

- The stable models of

$$2 \leq \{p, q, r\}^c \leq 2$$

are

$$\{p, q\}, \{p, r\}, \{q, r\}.$$

- Expressions of the forms

$$l \leq \{\ldots\}^c, \quad \{\ldots\}^c \leq u, \quad l \leq \{\ldots\}^c \leq u$$

can be used in LPARSE code in the $head$ of a rule, with both $\leq$ and the superscript c dropped:

$$l\ \{\ldots\}, \quad \{\ldots\}\ u, \quad l\ \{\ldots\}\ u.$$

LPARSE treatment of cardinality expressions

- LPARSE understands expressions of these types in different ways depending on whether they occur in the body or in the head of a rule.
- For example, LPARSE rules

```
r :- 1 {p,q}.
1 {p,q} :- r.
```

stand for

$$1 \leq \{p, q\} \rightarrow r$$

and

$$r \rightarrow 1 \leq \{p, q\}^c$$

respectively.

- A choice formula is included in the second case, but not in the first.

Variables in LPARSE

- A group of rules that follow a pattern can be often described concisely in LPARSE using schematic variables, for example, consider file var:
```
p(1..4).
#domain p(I).
q(I) :- not q(I-1).
```
- The first line is abbreviation for a group of 4 rules:
```
p(1). p(2). p(3). p(4).
```
It defines the auxiliary domain predicate p, which is used in the second line to declare I to be a variable with the domain $\{1, \ldots, 4\}$.
- The last line of var is interpreted as schematic representation of 4 rules (example of grounding):
```
q(1) :- not q(0).
q(2) :- not q(1).
q(3) :- not q(2).
q(4) :- not q(3).
```

Variables (ctd)

- LPARSE interprets var as the conjunction of formulas (15)
  $p(i)$,
  $\neg q(i-1) \rightarrow q(i)$
  $(1 \leq i \leq 4)$.

- In response to the command
  `% lparse var | smodels 0`
  SMODELS will compute the only stable model of this conjunction:
  `Stable Model: q(1) q(3) p(1) p(2) p(3) p(4)`

- The auxiliary atoms $p(1), \ldots, p(4)$ in the output can be suppressed by including the declaration
  `hide p(_).`

Grounding

- Domain predicates can be included directly in the bodies of the rules:
  ```
  p(1..4).
  q(I) :- p(I), not q(I-1).
  ```

- These two lines represent the conjunction of formulas (16)
  $p(i)$
  $p(i) \wedge \neg q(i-1) \rightarrow q(i)$
  $(1 \leq i \leq 4)$

- Since the conjunction of formulas (16) is intuitionistically equivalent to the conjunction of formulas (15), these two conjunctions have the same stable models.

Another use of variables

- Variables can be used to describe a list of literals that is formed according to a pattern. For example,

```
p(1..4).
2 {q(I) : p(I)} 3.
```

is shorthand for

```
p(1). p(2). p(3). p(4).
2 {q(1), q(2), q(3), q(4)} 3.
```

Graph Coloring

- An n-$coloring$ of a graph G is a function f from its set of vertices to $\{1, \ldots, n\}$ such that $f(x) \neq f(y)$ for every pair of adjacent vertices x, y.

- Task: use ASP to find an n-coloring of a given graph or to determine that it does not exist.

- Approach: Write a program whose answer sets are in a 1-1 correspondence with the n-colorings of G.

The program

- Let V be the set of vertices of the graph, and E the set of its edges. The program consists of the rules

  (17) $1 \leq \{color(x,1), \ldots, color(x,n)\}^c \leq 1 \qquad (x \in V)$,

  (18) $\leftarrow color(x,i), color(y,i) \qquad (\{x,y\} \in E;\ 1 \leq i \leq n)$.

- PROPOSITION 13.

  A set X of atoms is a stable model of the conjunction of (17) and (18) iff X is

  (19) $\{color(x, f(x)) : x \in V\}$

  for some n-coloring f *of* $\langle V,\ E\rangle$.

- (17) describes a superset of the set of n-colorings of G that we are trying to capture

  (18) consists of the constraints that weed out the bad elements of that superset

- This $generate\text{-}and\text{-}test$ organization is typical for simple ASP programs.

Proof of Proposition 13

- By Proposition 12(iii), each of the formulas (17) has n stable models $\{color(x, i)\}$ (i = 1, . . . , n).
- By Proposition 11, it follows that arbitrary stable models of the conjunction of these formulas are unions of such singletons, one per each $x \in V$.
- Stable models of (17) can be characterized as sets of the form (19), where f is a function from V to $\{1, \ldots, n\}$.
- By Proposition 4, it follows that the stable models of the conjunction of (17) with the constraints (18) can be characterized as the sets of the form (19) that do not satisfy the bodies of the constraints.
- The last condition can be expressed by saying that the equalities $f(x)$ = i and $f(y)$ = i cannot hold simultaneously when $\{x, y\} \in E$, which means that $f(x) \neq f(y)$ whenever $\{x, y\} \in E$.

The program in LPARSE

- Program (17), (18) can be encoded in LPARSE as

```
c(1..n).
1 {color(X,I) : c(I)} 1 :- v(X).
:- color(X,I), color(Y,I), e(X,Y), c(I).
```

- The domain predicates v and e are defined in a separate file, for instance

```
v(0..7).

e(0,1). e(1,2). e(2,3). e(3,0).
e(4,5). e(5,6). e(6,7). e(7,4).
e(0,4). e(1,5). e(2,6). e(3,7).
```

- If n= 2, SMODELS produces the set of atoms describing a 2-coloring

```
Stable Model: color(0,1) color(1,2) color(2,1) color(3,2)
color(4,2) color(5,1) color(6,2) color(7,1)
```

Cliques

- A $clique$ in a graph G is a set of pairwise adjacent vertices of G.
- Task: use ASP to find a clique of cardinality $\geq n$ in a given graph or to determine that it does not exist.
- Approach: write a program whose answer sets are in a 1-1 correspondence with cliques of cardinalities $\geq n$.

The Program

- Let V be the set of vertices of the graph, and E the set of its edges. The program consists of the rules

  (20) $n \leq \{in(x) : x \in V\}^c$,

  (21) $\leftarrow in(x), in(y)$ $\qquad\qquad$ $(x, y \in V; x \neq y; \{x, y\} \notin E)$.

- PROPOSITION 14.

  A set X of atoms is a stable model of the conjunction of (20) and (21) iff X is

  (22) $\{in(x) : x \in C\}$

  for some clique C in G such that $|C| \geq n$.

Proof of Proposition 14

- By Proposition 12(i), the stable models of (20) can be characterized as sets of the form (22), where C is a set of vertices of a cardinality $\geq n$.
- By Proposition 4, it follows that the stable models of the conjunction of (20) with the constraints (21) can be characterized as the sets of the form (22) that do not satisfy the bodies of the constraints.
- The last condition can be expressed by saying that the conditions $x \in C$ and $y \in C$ cannot hold simultaneously for two different non-adjacent vertices x, y, which means that C is a clique.

The program in LPARSE

- Program (20), (21) can be encoded in LPARSE as

```
n {in(X) : v(X)}.
:- in(X), in(Y), v(X;Y), X!=Y, not e(X,Y), not e(Y,X).
```

- The domain predicates v and e are assumed to characterize the vertices and edges of G.

- $v(X;Y)$ is an LPARSE abbreviation for $v(X), v(Y)$, and `!=` represents $\neq$.

- Condition $not\ e(X,Y),\ not\ e(Y,X)$ expresses that X and Y are non-adjacent.

Schur Numbers

- A set S of integers is called $sum\text{-}free$, if there are no numbers x, y in S such that x+y is in S.
- Example: $\{1, 3, 5\}$ is sum-free, and $\{2, 3, 5\}$ and $\{2, 4\}$ are not.
- Task: use ASP to find, for given k and n, a partition of the interval $\{1, \ldots, n\}$ into at most k sum-free sets or to determine that such a partition does not exist.
- The largest n such that $\{1, \ldots, n\}$ can be partitioned into k sum-free set is called the k-th $Schur\ number$ and denoted by $S(k)$.

The program

- The atoms $s_i(x)$ ($1 \leq i \leq k$, $1 \leq x \leq n$) are used to express that x belongs to the i-th set S_i in a partition of $\{1, \ldots, n\}$ into sum-free sets $S_1, \ldots, S_k$:

  (23) $1 \leq \{s_1(x), \ldots, s_k(x)\}^c \leq 1 \quad (1 \leq x \leq n)$,

  (24) $\leftarrow s_i(x), s_i(y), s_i(x + y) \quad (1 \leq i \leq k; x, y \geq 1; x + y \leq n)$.

- PROPOSITION 15.

  A set X of atoms is a stable model of the conjunction of (23) and (24) iff X is

  (25) $\{s_i(x) : 1 \leq i \leq k; x \in S_i\}$

  for sum-free pairwise disjoint sets $S_1, \ldots, S_k$ such that

  (26) $S_1 \cup \ldots \cup S_k = \{1, \ldots, n\}$.

Proof of Proposition 15

- By Proposition 12(iii), each of the formulas (23) has k stable models $\{s_i(x)\}$ ($i = 1, \ldots, k$).
- By Proposition 11, it follows that arbitrary stable models of the conjunction of these formulas are unions of such singletons, one per each $x \in \{1, \ldots, n\}$.
- That is, the stable models of (23) can be characterized as sets of the form (25), where the sets S_i are pairwise disjoint and satisfy (26).
- By Proposition 4, it follows that the stable models of the conjunction of (23) with the constraints (24) can be characterized as sets of the form (25), where S_i are pairwise disjoint, satisfy (26), and do not satisfy the bodies of the constraints.
- The last condition can be expressed by saying that each S_i is sum-free.

The program in LPARSE

- The rules (23) and (24) can be written in LPARSE as:

```
subset(1..k).
number(1..n).
#domain number(X;Y).

1 {s(I,X) : subset(I)} 1.
:- s(I,X), s(I,Y), s(I,X+Y), subset(I), X+Y<=n.
```

- The output produced by SMODELS is:

```
Stable Model: s(3,1) s(1,2) s(1,3) s(3,4) s(2,5) s(2,6) s(2,7)
s(2,8) s(2,9) s(3,10) s(1,11) s(1,12) s(3,13)
```

which represents a partition of $\{1, \ldots, 13\}$ into 3 sum-free sets:

$$\{2, 3, 11, 12\} \cup \{5, 6, 7, 8, 9\} \cup \{1, 4, 10, 13\}$$

Tiling

- Task: use ASP to find a way to cover an 8 x 8 chessboard by twenty-one 3 x 1 tiles and one 1 x 1 tile.
- Problem reformulated: place twenty-one 3 x 1 tiles on an 8 x 8 chessboard without overlaps.
- Horizontally placed tile: $h(x, y)$ $(0 \leq x \leq 5, 0 \leq y \leq 7)$, where x and y are the coordinates of the tile's southwest corner.
- Vertically placed tile: $v(x, y)$ $(0 \leq x \leq 7, 0 \leq y \leq 5)$, where x and y are the coordinates of the tile's southwest corner.
- Call these 96 atoms $A_1, \ldots, A_{96}$

The program

- The stable models of the rule

  (27) $21 \leq \{A_1, \ldots, A_{96}\}^c \leq 21$

  correspond to all possible ways to place 21 tiles on the chessboard.

- Overlaps between two horizontal tiles are eliminated by the rules

  (28) $\leftarrow h(x, y), h(x + i, y) \quad (0 \leq x, y \leq 7;\ i = 1, 2)$.

- Overlaps between two vertical tiles are eliminated by the rules

  (29) $\leftarrow v(x, y), v(x, y + i) \quad (0 \leq x, y \leq 7;\ i = 1, 2)$.

- Overlaps between a horizontal tile and a vertical tile are eliminated by

  (30) $\leftarrow h(x, y), v(x + i, y - j) \quad (0 \leq x, y \leq 7; 0 \leq i, j \leq 2)$.

- The stable models of program (27)-(30) correspond to the solutions to the tiling problem we are interested in.

The program in LPARSE

- The program (27)-(30) can be represented in the language of LPARSE as:

```
number(0..7).
#domain number(X;Y;I;J).

hpos(X,Y) :- X<=5.
vpos(X,Y) :- Y<=5.

21 {h(XX,YY) : hpos(XX,YY), v(XX,YY) : vpos(XX,YY)} 21.

:- h(X,Y), h(X+I,Y), 0<I, I<=2.
:- v(X,Y), v(X,Y+I), 0<I, I<=2.
:- h(X,Y), v(X+I,Y-J), I<=2, J<=2.
```

- The output of SMODELS is:

```
Stable Model: h(5,1) h(5,0) h(3,7) h(3,6) h(3,5) h(3,4) h(3,3)
h(3,2) h(2,1) h(2,0) h(0,7) h(0,6) v(7,5) v(7,2) v(6,5) v(6,2)
v(2,3) v(1,3) v(1,0) v(0,3) v(0,0)
```

Hamiltonian Cycles

- A $Hamiltoanian\ cycle$ in a directed graph G is a closed path that passes through each vertex of G exactly once.
- Task: use ASP to find a Hamiltonian cycle in a given directed graph or to determine that it does not exist.
- Atoms $in(x, y)$ for all edges $\langle x, y\rangle$ of G are used to express that $\langle x, y\rangle$ belongs to the path.

The program

- The generate part of the program consists of the choice rules

  (31) $\{in(x,y)\}^c \quad (\langle x,y\rangle \in E)$

- The constraints that eliminate all subsets of E other than Hamiltonian cycles are

  (32) $\leftarrow 2 \leq \{in(x,y\} : y \in A_x\} \quad (x \in V)$,

  where A_x stands for $\{y : \langle x,y\rangle \in E\}$, and

  (33) $\leftarrow 2 \leq \{in(x,y) : x \in B_y\} \quad (y \in V)$,

  where B_y stands for $\{x : \langle x,y\rangle \in E\}$.

- Every vertex of G should be reachable by a sequence of in-edges from some fixed vertex v_0.

The program (ctd)

- Auxiliary atom $r(x)$ ("x is reachable from x_0") is defined as

  (34) $r(x) \leftarrow in(x_0, x) \quad (x \in V)$,

  (35) $r(y) \leftarrow r(x), in(x, y) \quad (\langle x, y \rangle \in E)$

- The reachability constraints are:

  (36) $\leftarrow not\ r(x) \quad (x \in V)$.

- Program has generate part (31), test part (32), (33), (36), and define part (34), (35). This "generate-define-test" structure is typical for more advanced ASP programs.

Proposition 16

- *A set X of atoms is the essential part of a stable model of (31)-(36) iff X has the form*

  (37) $\{in(x, y) : \langle x, y\rangle \in H\}$

  where H is the set of edges of a Hamiltonian cycle in G. Furthermore, different stable models of this program have different essential parts.

- For any set $H \subseteq R$, by R_H we denote the set of atoms $r(x)$ for all vertices x to which there is a path of nonzero length from x_0 over edges in H.

- LEMMA 17.
  A set X of atoms is a stable model of the conjunction of formulas (31), (34) and (35) iff X is

  (38) $\{in(x, y) : \langle x, y\rangle \in H\} \cup R_H$

  for some subset H of E.

Proof of Lemma 17

- Denote the conjunction of formulas (31) by F, and the conjunction of formulas (34), (35) by G.
- By Theorem 8, X is a stable model of $F \wedge G$ iff there exists a stable model $\{A_1, \ldots, A_n\}$ of F such that X is a stable model of $A_1 \wedge \ldots \wedge A_n \wedge G$.
- By Proposition 3, it follows that the stable models of $F \wedge G$ can be characterized as the stable models of formulas of the form

  (39) $$\bigwedge_{\langle x,y\rangle \in H} in(x,y) \wedge G.$$

  for arbitrary subsets H of E.
- Formula (39) is a Horn formula, and its minimal model is its only stable model (by Theorem 2).
- It remains to observe that the minimal model of (39) is (38).

Proof of Proposition 16

- A set $H \subseteq E$ is the set of edges of a Hamiltonian cycle in G iff it satisfies the following conditions:

  (i) H does not contain two different edges leaving the same vertex.
  (ii) H does not contain two different edges ending at the same vertex.
  (iii) For every vertex x of G, there exists a path of nonzero length from x_0 to x over edges in H.

- By Lemma 17 and Proposition 4, a set X of atoms is a stable model of program (31)-(36) iff X has the form (38), where $H \subseteq E$, and does not satisfy the bodies of the constrains (32), (33), (36).

- It is clear that
  - (i) holds iff (38) does not satisfy the bodies of constraints (32);
  - (ii) holds iff (38) does not satisfy the bodies of constraints (33);
  - (iii) holds iff (38) does not satisfy the bodies of constraints (36).

- Therefore, X is a stable model of (31)-(36) iff X has the form (38) for a subset H of E satisfying conditions (i)-(iii). Both parts of the statement of Proposition 16 now follow, because the essential part of (38) is (37).

The program in LPARSE

- Assuming that x_0 is 0, program (31)-(36) can be represented in the language of LPARSE as

```
{in(X,Y)} :- e(X,Y).

:- 2 {in(X,Y) : e(X,Y)}, v(X).
:- 2 {in(X,Y) : e(X,Y)}, v(Y).

r(X) :- in(0,X), v(X).
r(Y) :- r(X), in(X,Y), e(X,Y).

:- not r(X), v(X).

hide r(_).
```

The Blocks World

- The blocks world consists of several blocks $1, \ldots, n$, placed on the table so that they form a tower or several towers. Blocks can be moved around.

- Task: use ASP to find a sequence of actions that takes the blocks world from a given initial state to a given goal state.

- Approach: write ASP program that represents the set of all possible configurations of n blocks.

- Positions of the blocks are described by the atoms $on(x, y)$, where $x \in \{1, \ldots, n\}, y \in \{1, \ldots, n, table\}, x \neq y$.

The program

- Choose arbitrarily, for each block x, a unique location:

  (40) $1 \leq \{on(x, y) : y \in \{1, \ldots, n, table\} \setminus \{x\}\}^c \leq 1$

  $(1 \leq x \leq n)$.

- Do not allow two blocks to be on top of the same block:

  (41) $\leftarrow 2 \leq \{on(x, y) : x \in \{1, \ldots, n\} \setminus \{y\}\}$

  $(1 \leq y \leq n)$.

- Atoms $s(x)$, where $1 \leq x \leq n$ will express that x is supported by the table:

  (42) $s(x) \leftarrow on(x, table) \quad (1 \leq x \leq n)$,

  (43) $s(x) \leftarrow s(y), on(x, y) \quad (1 \leq x, y \leq n; x \neq y)$.

- The absence of blocks floating in space is expressed by the constraints:

  (44) $\leftarrow not\ s(x) \quad (1 \leq x \leq n)$.

The program in LPARSE

- Program (40)-(44) can be expressed in LPARSE by:

```
block(1..n).

1 {on(X,Y) : block(Y) : X != Y, on(X,table)} 1 :- block(X).

:- 2 {on(X,Y) : block(X) : X != Y}, block(Y).

s(X) :- on(X,table), block(X).
s(X) :- s(Y), on(X,Y), block(X;Y), X != Y.

:- not s(X), block(X).

hide s(_).
```

Strong Negation

- Assume two kinds of atoms: $positive$ and $negative$. Each negative atom is an expression of the form $\sim A$, where A is a positive atom.
- The symbol $\sim$ is called strong negation (or "classical", or "true").
- A set of atoms is $coherent$ if it does not contain "complementary" pairs of atoms A, $\sim A$.
- Consider the program

  (45) $\{p\}^c$,
  q,
  $\sim q \leftarrow \neg p$.

- It has two positive atoms p, q and one negative atom $\sim q$.
- The stable models are $\{p, q\}$ and $\{q, \sim q\}$. The first one is coherent, and the second is not.

Proposition 18

- *A set X of atoms is a coherent stable model of a formula F iff X is a stable model of the formula*

  (46) $F \wedge \bigwedge_{A} \neg(A \wedge \sim A)$,

  where the big conjunction extends over all positive atoms A such that both A and $\sim A$ are head atoms of F.

- PROOF.

  By proposition 4, X is a stable model of (46) iff X is a stable model of F which does not have subsets of the form $\{A, \sim A\}$ such that $A, \sim A$ are head atoms of F.

  By Theorem 1, this condition on X is equivalent to saying that X is a coherent stable model of F.

Strong negation in LPARSE

- In LPARSE, strong negation is written as $-$ and LPARSE should be called with the option **- -true-negation.**

- SMODELS generate only coherent answer sets. For example,

```
{p}.
q.
-q :- not p.
```

will have only one model:

```
Stable Model: p q
```

The purpose of Strong negation in ASP

- Strong negation allows us to distinguish between the assertions
  (1) ”A is false”
  (2) ”A is not known to be true”
- Statement (1) is expressed by the presence of the negative atom $\sim A$ in a coherent stable model. Statement (2) is expressed by the absence of the positive atom A, which is a weaker condition.
- The rule

  $\sim A \leftarrow not\ A$

  (”A is false if there is no evidence to the contrary”) is an ASP representation of the closed world assumption for the positive atom A.
- The rule

  $A \leftarrow not\ \sim A$

  expresses the inverse closed world assumption: A is true if there is no evidence to the contrary.

Proposition 19

- *Let F be a formula and A a positive atom such that $\sim A$ does not occur in F. For any set X of atoms, X is a coherent stable model of*

  (48) $F \wedge (\neg A \rightarrow \sim A)$

  iff

  (i) X is a stable model of F and $A \in X$, or

  (ii) $X = Y \cup \{\sim A\}$, where Y is a stable model of F such that $A \notin Y$.

Proof of Proposition 19

- By Theorem 8, X is a stable model of (48) iff X is a stable model of a formula of the form

  (49) $A_1 \wedge \ldots \wedge A_n \wedge (\neg A \rightarrow\!\sim A)$,

  where $\{A_1, \ldots, A_n\}$ is a stable model of F.

- Case 1: A equals one of the atoms A_i.

  Then (49) is intuitionistically equivalent to $A_1 \wedge \ldots \wedge A_n$, and X is a stable model of (49) iff $X = \{A_1, \ldots, A_n\}$.

- Case 2: A is different from all atoms A_i.

  Then A is not a head atom of (49). By Proposition 9, it follows that X is a stable model of (49) iff X is a stable model of the formula

  $$A_1 \wedge \ldots \wedge A_n \wedge (\neg\bot \rightarrow\!\sim A),$$

  which is intuitionistically equivalent to

  $$A_1 \wedge \ldots \wedge A_n \wedge \sim A.$$

  So X is a stable model of (48) iff $X = Y \cup \{\sim A\}$, where Y stands for $\{A_1, \ldots, A_n\}$.

Planning in The Blocks World

- Task: use ASP to find a sequence of $sets$ of actions that takes the blocks world from a given initial state to a state satisfying a given goal condition.

- There are n^2 possible actions, where n is the number of blocks: any block $x \in \{1, \ldots, n\}$ can be moved to any location $l \in \{1, \ldots, n, table\}$ different from x.

- Assumptions:
  - a block can be moved only when there are no blocks on top of it;
  - at most k actions can be executed concurrently;
  - a block x can be moved onto a block y only if y is not being moved at the same time.

Histories

- A $history$ is a finite sequence

$$s_0, e_0, s_1, e_1, \ldots, e_{m-1}, s_m$$

  where $s_0, s_1, \ldots, s_m$ are states of the blocks world, and each e_i $(0 \leq i \leq m)$ is a set of actions, which, when executed concurrently in state s_i lead to state s_{i+1}.

- Histories will be described by:

  - the atoms $on(x, l, i)$ $(x \in \{1, \ldots, n\}, l \in \{1, \ldots, n, table\}, x \neq l, i \in \{0, \ldots, m\})$, expressing that x is on l in state s_i, and

  - the atoms $move(x, l, i)$ $(x \in \{1, \ldots, n\}, l \in \{1, \ldots, n, table\}, x \neq l, i \in \{0, \ldots, m-1\})$, expressing that x is moved onto l as part of event e_i.

- Approach: Write a program whose stable models represent the histories with a given initial state s_0 and a given length m such that their final state s_m satisfies a given goal condition.

The program

- Each event e_i can be composed of up to k actions, chosen arbitrarily:

  (51) $\{move(x, l, i) : 1 \leq x \leq n,\ l \in \{1, \ldots, n, table\},\ x \neq l\}^c \leq k$

  $(1 \leq i < m)$

- A block can be moved only if it is clear:

  (52) $\leftarrow move(x, l, i),\ on(y, x, i)$

  $(1 \leq x, y \leq n,\ l \in \{1, \ldots, n, table\},\ x \neq l,\ x \neq y,\ 0 \leq i < m)$

- A block can be moved if the destination is not a block that is being moved also:

  (53) $\leftarrow move(x, y, i), move(y, l, i)$

  $(1 \leq x, y \leq n,\ l \in \{1, \ldots, n, table\},\ x \neq y,\ y \neq l,\ 0 \leq i < m)$

The program (ctd)

- The following rules define the locations of blocks in state s_i in terms of their initial locations and the events $e_0, \ldots, e_{i-1}$

  (54) $on(x, init(x), 0)$

  $(1 \leq x \leq n)$, where $init(x)$ stands for the initial location of x

  (55) $on(x, l, i+1) \leftarrow move(x, l, i)$

  $(1 \leq x \leq n),\ l \in \{1, \ldots, n, table\},\ x \neq l,\ 0 \leq i < m)$

- The next rule expresses the uniqueness of the location of a block using strong negation:

  (56) $\sim on(x, l, i) \leftarrow on(x, l', i)$

  $(1 \leq x \leq n),\ l, l' \in \{1, \ldots, n, table\}, x \neq l,\ x \neq l',\ l \neq l',\ 0 \leq i \leq m)$

The program (ctd)

- The commonsense law of inertia for the blocks world:

  (57) $on(x, l, i + 1) \leftarrow on(x, l, i), not\ \sim on(x, l, i + 1)$

  $(1 \leq x \leq n,\ l \in \{1, \ldots, n, table\},\ x \neq l,\ 0 \leq i < m)$

- The following constraints express that $s_1, \ldots, s_m$ are valid states of the blocks world, and that s_m satisfies the goal condition G:

  (58) $\leftarrow 2 \leq \{on(x, y, i) : x \in \{1, \ldots, n\} \backslash \{y\}\}$

  $(0 \leq y \leq n, 0 \leq i < m)$;

  (59) $\leftarrow not\ G$.

The program in LPARSE

```
step(0..m).
block(1..n).
location(1..n;table).

#domain step(I).
#domain block(X;Y;Z).
#domain location(L;L1).

{move(XX,LL,I) : block(XX) : location(LL) : XX !=LL} k :- I < m.

:- move(X,L,I), on(Y,X,I), X != L, X != Y, I < m.
:- move(X,Y,I), move(Y,L,I), X != Y, Y != L, I < m.

on(X,L,0) :- init(X,L).
on(X,L,I+1) :- move(X,L,I), X != L, I < m.
-on(X,L,I) :- on(X,L1,I), X != L, X != L1, L != L1.
on(X,L,I+1) :- on(X,L,I), not -on(X,L,I+1), X != L, I < m.
```

The program in LPARSE (ctd)

```
:- 2 {on(XX,Y,I) : block(XX) : XX != Y}.
:- not goal.

hide.
show move(_,_,_).
```

The initial state and the goal condition are assumed to be defined in a separate file, for example:

```
init(1,2). init(2,table). init(3,4).
init(4,table). init(5,6). init(6,table).

goal :- on(2,1,m), on(3,2,m), on(6,5,m), on(5,4,m).
```

Proofs of Theorems

- LEMMA 20.
  If $X \models F$ and a set Y contains all head atoms of F then $X \cap Y \models F^X$.

- PROOF (by structural induction on F).
  Assume that $X \models F$. Clearly F is not $\bot$.

- Case 1: F is an atom A. Since $X \models F$, F^X is A and $A \in X$. Since A is a head atom, we can conclude that $A \in X \cap Y$.

- Case 2: F is $G \wedge H$. Since $X \models F$, we know that F^X is $G^X \wedge H^X$, $X \models G$ and $X \models H$. Since all head atoms of G and H belong to Y, from the induction hypothesis we conclude that $X \cap Y \models G^X$ and $X \cap Y \models H^X$. Therefore, $X \cap Y \models F^X$.

- Case 3: F is $G \vee H$. Similar to case 2.

- Case 4: F is $G \rightarrow H$. Since $X \models F$, F^X is $G^X \rightarrow H^X$.
  Case 4.1: $X \models G$. Then $X \models H$. Since all head atoms of H belong to Y, from the induction hypothesis $X \cap Y \models H^X$. Therefore $X \cap Y \models F^X$.
  Case 4.2: $X \not\models G$. Then G^X is $\bot$, so that F^X is tautology.

Proof of Theorem 1

- THEOREM 1.

  Any stable model of F is a subset of the set of head atoms of F.

- Let X be a stable model of F, and Y the set of head atoms of F.
- By Lemma 20, $X \cap Y \models F^X$.
- Since X is minimal among the sets satisfying F^X,

  it follows that $X \cap Y = X$.

  Therefore $X \subseteq Y$.

LEMMA 21 and its proof

- LEMMA 21.
  For any Horn formula F and any two sets X and Y of atoms, if $X \subseteq Y$ and $Y \models F$ then $X \models F$ iff $X \models F^Y$.

- PROOF.
  1) Assume that F is a single implication

  (60) $A_1 \wedge \ldots \wedge A_n \rightarrow A$.

  Case 1: $A_1, \ldots, A_n$ belong to Y. Under the assumption $Y \models F$, the consequent A of F belongs to Y also, so that $F^Y = F$.

  Case 2: For some i, $A_i \notin Y$. Under the assumption $X \subseteq Y$, $A_i \notin X$, so that X satisfies F. On the other hand, F^Y is the tautology $\bot \rightarrow A^Y$, so that X satisfies F^Y as well.

  2) If F is a conjunction $F_1 \wedge \ldots \wedge F_m$ of several implications of the form (60), then X satisfies F iff X satisfies each F_j. Under the assumption $Y \models F$, F^Y is $F_1^Y \wedge \ldots \wedge F_m^Y$. Therefore, X satisfies F^Y iff X satisfies each of the conjunctive terms F_j^Y. The assertion of the lemma follows from the special case proved above.

Proof of Theorem 2

- THEOREM 2.
  For any Horn formula F, the minimal model of F is the only stable model of F.
- PROOF.
  Let M be the minimal model of a Horn formula F.
- Lemma 21, applied to M as Y, shows that F^M is satisfied by M but is not satisfied by any proper subset of M.
- Therefore, M is a stable model of F.
- Now take any stable model Y of F. By the choice of M, $M \subseteq Y$.
- Lemma 21, applied to M as X shows that $M \models F^Y$.
- By definition of stable model, Y is minimal among the sets satisfying F^Y.
- Therefore, $Y \subseteq M$.
- We have proved that $Y = M$.

Lemma 22, Lemma 23 and their proofs

- LEMMA 22.
  For any formula F and any set X of atoms, $X \models F^X$ iff $X \models F$.
- PROOF.
  Reduct F^X is obtained from F by replacing some subformulas that are not satisfied by X with $\bot$.
- LEMMA 23.

  For any two formulas F and G and any set X of atoms,

  (a) $(F \wedge G)^X$ is equivalent to $F^X \wedge G^X$ in classical logic, and

  (b) $(F \vee G)^X$ is equivalent to $F^X \vee G^X$ in classical logic.
- PROOF.
  Part (a): consider two cases, depending on whether X satisfies $F \wedge G$.
  If it does, then the two formulas are equal to each other;
  if not, then each of them is equivalent to $\bot$.

  Part(b): the proof is similar.

Lemma 24 and its proof

- LEMMA 24.
  For any formula F and any two sets X and Y of atoms, $X \models F^Y$ iff $\langle X \cap Y, Y \rangle \models F$.

- PROOF: (by structural induction on F)

  - If F is $\bot$, then the assertion of the lemma is trivial.
  - If F is an atom A,

$$\begin{aligned} X \models A^Y \text{ iff } & A \in Y \text{ and } A \in X \\ \text{iff } & A \in X \cap Y \\ \text{iff } & \langle X \cap Y, Y \rangle \models A. \end{aligned}$$

  - If F is $G \wedge H$, then, using Lemma 23(a),

$$\begin{aligned} X \models (G \wedge H)^Y \text{ iff } & X \models G^Y \wedge H^Y \\ \text{iff } & X \models G^Y \text{ and } X \models H^Y \\ \text{iff } & \langle X \cap Y, Y \rangle \models G \text{ and } \langle X \cap Y, Y \rangle \models H \\ \text{iff } & \langle X \cap Y, Y \rangle \models G \wedge H. \end{aligned}$$

Proof of Lemma 24 (ctd)

- If F is $G \vee H$, then the reasoning is similar, using Lemma 23(b).
- If F is $G \rightarrow H$,

  $X \models (G \rightarrow H)^Y$

  iff $Y \models G \rightarrow H$ and $X \models G^Y \rightarrow H^Y$

  iff $Y \models G \rightarrow H$ and $X \not\models G^Y$ or $X \models H^Y$

  iff $Y \models G \rightarrow H$ and $\langle X \cap Y, Y \rangle \not\models G$ or $\langle X \cap Y, Y \rangle \models H$

  iff $\langle X \cap Y, Y \rangle \models G \rightarrow H$.

Lemma 25 and its proof

- LEMMA 25.
  Let F, G, F', G' be formulas such that G' is obtained from F' by replacing some (zero or more) occurrences of F with G. For any set X of atoms, if F^X is equivalent to G^X, then $(F')^X$ is equivalent to $(G')^X$.
- PROOF:
  Assume that F^X is equivalent to G^X. By Lemma 22, it follows that
  (62) $X \models F \leftrightarrow G$.
- We will prove that $(F')^X$ is equivalent to $(G')^X$ by structural induction on F'.
- This assertion is trivial when F' equals F and also when the number of occurrences of F in F' that are being replaced is 0. In particular, the cases when F' is $\bot$ or an atom are trivial (BASE CASE).
- Assume that F' has the form $F'_1 \odot F'_2$ and G' is $G'_1 \odot G'_2$, where G'_i is obtained from F'_i by replacing some occurrences of F with G.
- Case 1: $X \not\models F'$. In view of (61), $X \not\models G'$, so that $(F')^X = \bot$ and $(G')^X = \bot$.
- Case 2: $X \models F'$. In view of (61), $X \models G'$, so that $(F')^X = (F'_1)^X \odot (F'_2)^X$ and $(G')^X = (G'_1)^X \odot (G'_2)^X$, and the claim follows by the induction hypothesis.

Combined statement of Theorems 5 and 7

For any formula F and G, the following conditions are equivalent:

(i) F is strongly equivalent to G,

(ii) for every unary formula H, $F \wedge H$ and $G \wedge H$ have the same stable models,

(iii) F is equivalent to G in the logic of here-and-there,

(iv) for any set X of atoms, F^X is equivalent to G^X in classical logic.

Proof of Theorems 5 and 7

- From (i) to (ii): obvious.

- From (ii) to (iii): assume that F is not equivalent to G in the logic of here-and-there, and let $\langle X, Y\rangle$ be an HT-interpretation that satisfies F, but not G. Then $X \subseteq Y$, and by Lemma 24, $X \models F^Y, X \not\models G^Y$. Since $X \models F^Y$, F^Y is not $\bot$, which implies that $Y \models F$. By Lemma 22, it follows that $Y \models F^Y$.
  - Case 1: $Y \not\models G^Y$. By Lemma 22, $Y \not\models G$, so that Y is not a stable model of $G \wedge H$ for any H. But if we take H to be $\bigwedge_{A \in Y} A$ then Y is a stable model of $F \wedge H$. Indeed, by Lemma 23(a), $(F \wedge H)^Y$ is equivalent to $F^Y \wedge H^Y$, which is the same as $F^Y \wedge H$; both conjunctive terms of this formula are satisfied by Y, but the second term is not satisfied by any proper subset of Y.

Proof of theorems 5 and 7 (ctd)

- Case 2: $Y \models G^Y$. Since $X \not\models G^Y$, X is different from Y; therefore, X is a proper subset of Y. Let H be the unary formula

$$\bigwedge_{A \in X} A \wedge \bigwedge_{A, A' \in Y \setminus X} (A \rightarrow A').$$

  Set Y is not a stable model of $F \wedge H$. As in Case 1, $(F \wedge H)^Y$ is equivalent to $F^Y \wedge H$; X is proper subset of Y that satisfies both conjunctive terms. We will show, on the other hand, that Y is a stable model of $G \wedge H$, which contradics condition (ii). In view of Lemma 23(a), $(G \wedge H)^Y$ is equivalent to $G^Y \wedge H$. Clearly Y satisfies both conjunctive terms; the only proper subset of Y that satisfies H is X, and X does not satisfy G^Y.

Proof of theorems 5 and 7 (ctd)

- From (iii) to (iv): if F and G are satisfied by the same HT-interpretations then, by Lemma 24, for any set Y of atoms, F^Y and G^Y are satisfied by the same sets of atoms.

- From (iv) to (i): immediate from Lemma 25.

Lemma 26 and its proof

- LEMMA 26.
  If X is a stable model of F then F^X is equivalent to $\bigwedge_{A \in X} A$.

- PROOF:
  Since all atoms occurring in these two formulas belong to X, it is sufficient to show that the formulas are satisfied by the same subsets of X. By the definition of a stable model, the only subset of X satisfying F^X is X.

Lemma 27 and its proof

- LEMMA 27.
  Let S be a set of atoms that contains all atoms occurring in a formula F but does not contain any head atoms of a formula G. For any set X of atoms, if X is a stable model of $F \wedge G$ then $X \cap S$ is a stable model of F.

- Since X is a stable model of $F \wedge G$, $X \models F$, so that $X \cap S \models F$, and by Lemma 22, $X \cap S \models F^{X \cap S}$. It remains to show that no proper subset Y of $X \cap S$ satisfies $F^{X \cap S}$.

- Let S' be the set of head atoms of G, and let Z be $X \cap (S' \cup Y)$.

Proof of Lemma 27 (ctd)

- CLAIM: Set Z has the following properties:

  (i) $Z \cap S = Y$;

  (ii) $Z \subset X$;

  (iii) $Z \models G^X$.

- To prove (i), note that since S' is disjoint from S, and Y is a subset of $X \cap S$,

$$Z \cap S = X \cap (S' \cup Y) \cap S = X \cap Y \cap S = (X \cap S) \cap Y = Y.$$

- To prove (ii), note that set Z is clearly a subset of X. It cannot be equal to X, because otherwise we would have, by (i),

  $Y = Z \cap S = X \cap S$;

  this is impossible, because Y is a proper subset of $X \cap S$.

- Property (iii) follows from Lemma 20, because $X \models G$, and $S' \cup Y$ contains all head atoms of G.

Proof of Lemma 27 (ctd)

- Since X is a stable model of $F \wedge G$, from property (ii), we can conclude that $Z \not\models (F \wedge G)^X$.
- Therefore, by Lemma 23(a) and property (iii), $Z \not\models F^X$. Since all atoms occurring in F belong to S, $F^X = F^{X \cap S}$, so that we can rewrite this formula as $Z \not\models F^{X \cap S}$.
- By property (i), we conclude that $Y \not\models F^{X \cap S}$.

Proof of Theorem 8

- THEOREM 8.
  Let F and G be formulas such that F does not contain any head atoms of G. A set X of atoms is a stable model of $F \wedge G$ iff there exists a stable model $\{A_1, \ldots, A_n\}$ of F such that X is a stable model of

  (62) $A_1 \wedge \ldots \wedge A_n \wedge G$.

- PROOF:
  Take formulas F and G such that F does not contain any head atoms of G, and let S be the set of atoms occurring in F.

- Observe that if a set X of atoms is a stable model of a formula of the form (62), where $A_1, \ldots, A_n \in S$, then $X \cap S = \{A_1, \ldots, A_n\}$. Indeed, by Lemma 27 with $A_1 \wedge \ldots \wedge A_n$ as F, $X \cap S$ is a stable model of $A_1 \wedge \ldots \wedge A_n$, and the only stable model of this formula is $\{A_1, \ldots, A_n\}$.

Proof of Theorem 8 (ctd)

- Therefore, the assertion to be proved can be reformulated as follows: a set X of atoms is a stable model of $F \wedge G$ iff

  (i) $X \cap S$ is a stable model of F, and

  (ii) X is a stable model of $\bigwedge_{A \in X \cap S} A \wedge G$.

- If $X \cap S$ is not a stable model of F then X is not a stable model of $F \wedge G$ by Lemma 27.

- Suppose that $X \cap S$ is a stable model of F. Then, by Lemma 26, $F^{X \cap S}$ is equivalent to $\bigwedge_{A \in X \cap S} A$. Therefore, by Lemma 23(a),

  $(F \wedge G)^X \leftrightarrow F^X \wedge G^X \; = \; F^{X \cap S} \wedge G^X \leftrightarrow \bigwedge_{A \in X \cap S} A \wedge G^X$

  $= \; (\bigwedge_{A \in X \cap S} A)^X \wedge G^X \leftrightarrow (\bigwedge_{A \in X \cap S} A \wedge G)^X.$

- We can conclude that X is a stable model of $F \wedge G$ iff X is a stable model of $\bigwedge_{A \in X \cap S} A \wedge G$.

Conclusion

- Many publications in the area of ASP are directed toward practical applications.
- The areas of science and technology where ASP may be useful are remarkably diverse.
- Success would have been impossible without efficient, reliable, carefully crafted answer set solvers.
- The main topic of this paper is theoretical.
- ASP is based on interesting mathematics, including some ideas developed in the early days of modern logic.
- Intuitionistic logic plays an important role in this theory.

Appendix A: Propositional Logic

- $(Propositional)\ formulas$ are formed from propositional atoms and the 0-place connective $\bot$ using the binary connectives $\wedge$, $\vee$, and $\rightarrow$.
- We use

$$\top \text{ as shorthand for } \bot \rightarrow \bot,$$
$$\neg F \text{ as shorthand for } F \rightarrow \bot,$$
$$F \leftrightarrow G \text{ as shorthand for } (F \rightarrow G) \wedge (G \rightarrow F).$$

- Atoms and negated atoms are called $literals$.
- The relation $X \models F$ between a set X of atoms and a formula F is defined recursively:
  - for an atom A, $X \models A$ if $A \in X$;
  - $X \not\models \bot$;
  - $X \models F \wedge G$ if $X \models F$ and $X \models G$;
  - $X \models F \vee G$ if $X \models F$ or $X \models G$;
  - $X \models F \rightarrow G$ if $X \not\models F$ or $X \models G$.

Syntax and Semantics (ctd)

- If $X \models F$, then we way that X $satisfies$ F, or is a $model$ of F.
- A formula is $tautology$, if it is satisfied by every set of atoms.
- A formula F is $equivalent$ to a formula G if $F \leftrightarrow G$ is a tautology (or, equivalently, if F and G have the same models).
- An occurrence of an atom A in a formula F is $positive$ if the number of implications containing that occurrence in the antecedent is even, and $negative$ otherwise.
- For example, both occurrences of p in (63) are positive, and q, r are negative:

  (63) $((p \rightarrow q) \wedge r) \rightarrow p$.

- An occurrence of an atom A in a formula F is $strictly\ positive$ if it does not belong to the antecedent of any implication in F. For example, the second occurrence of p in (63) is strictly positive, and the first is not.
- Since $\neg F$ is shorthand for $F \rightarrow \bot$, no occurrence of an atom in a formula of the form $\neg F$ can be strictly positive.

Logic of Here-and-There

- A 3-valued logic; originally proposed by Arend Heyting (inventor of intuitionistic logic) as a technical tool for the purpose of proving that intuitionistic logic is weaker than classical.
- His truth values can be interpreted as follows:
  - 0 denotes a correct proposition
  - 1 denotes a false proposition
  - 2 denotes a proposition that cannot be false but whose correctness is not proved
- We will identify a function from the set of atoms to the extended set of truth values $\{0, 1, 2\}$ with the ordered pair consisting of the set X of atoms that are mapped to 0 and the set Y of atoms that are mapped to 0 or 2.
- If an atom belongs to X then it is true $here$; if an atom belongs to Y then it is true $there$.
- An $HT\text{-}interpretation$ is an ordered pair $\langle X, Y \rangle$ of sets of atoms such that $X \subseteq Y$.

Logic of Here-and-There (ctd)

- The $satisfaction$ relation $\models$ between an HT-interpretation $\langle X, Y\rangle$ and a formula F is defined recursively:
  - for an atom A, $\langle X, Y\rangle \models A$ if $A \in X$;
  - $\langle X, Y\rangle \not\models \bot$;
  - $\langle X, Y\rangle \models F \wedge G$ if $\langle X, Y\rangle \models F$ and $\langle X, Y\rangle \models G$;
  - $\langle X, Y\rangle \models F \vee G$ if $\langle X, Y\rangle \models F$ or $\langle X, Y\rangle \models G$;
  - $\langle X, Y\rangle \models F \rightarrow G$ if
    (i) $\langle X, Y\rangle \not\models F$ or $\langle X, Y\rangle \models G$, and
    (ii) $Y \models F \rightarrow G$.
- A formula is $valid\ in\ the\ logic\ of\ here\text{-}and\text{-}there$ if it is satisfied by every HT-interpretation.
- A formula F is $equivalent$ to a formula $G\ in\ the\ logic\ of\ here\text{-}and\text{-}there$ if $F \leftrightarrow G$ is valid in the logic of here-and-there (or, equivalently, if F and G are satisfied by the same HT-interpretations).

Logic of Here-and-There

- Facts that relate the satisfaction relation of the logic of here-and-there to the satisfaction relation of classical logic:

  (64) $\langle X, X\rangle \models F$ iff $X \models F$.

  (65) If $\langle X, Y\rangle \models F$ then $Y \models F$.

  (66) $\langle X, Y\rangle \models \neg F$ iff $Y \models \neg F$.

- From (64): a formula can be valid in the logic of here-and-there only if it is a tautology.

- Two formulas can be equivalent to each other in the logic of here-and-there only if they are classically equivalent.

- Difference: $\neg\neg p$ is not equivalent to p in the logic of here-and-there. Indeed, by (66), the HT-interpretation $\langle \emptyset, \{p\}\rangle$ satisfies $\neg\neg p$, but it does not satisfy p.

Natural Deduction

- In the natural deduction system for propositional logic, the derivable objects are $sequents$- expressions of the form $\Gamma \Rightarrow F$, where F is a formula and Γ is a finite set of formulas.

- The axiom schemas are

  (67) $F \Rightarrow F$

  and

  (68) $\Rightarrow F \vee \neg F$.

- The latter is called $the\ law\ of\ excluded\ middle$.

Inference rules

- The introduction rules are:

  $(\wedge I)$ $\frac{\Gamma \Rightarrow F \quad \Delta \Rightarrow G}{\Gamma, \Delta \Rightarrow F \wedge G}$

  $(\vee I)$ $\frac{\Gamma \Rightarrow F}{\Gamma \Rightarrow F \vee G} \quad \frac{\Gamma \Rightarrow G}{\Gamma \Rightarrow F \vee G}$

  $(\rightarrow I)$ $\frac{\Gamma, F \Rightarrow G}{\Gamma \Rightarrow F \rightarrow G}$

- The elimination rules are:

  $(\wedge E)$ $\frac{\Gamma \Rightarrow F \wedge G}{\Gamma \Rightarrow F} \quad \frac{\Gamma \Rightarrow F \wedge G}{\Gamma \Rightarrow G}$

  $(\vee E)$ $\frac{\Gamma \Rightarrow F \vee G \quad \Delta_1, F \Rightarrow H \quad \Delta_2, G \Rightarrow H}{\Gamma, \Delta_1, \Delta_2 \Rightarrow H}$

  $(\rightarrow E)$ $\frac{\Gamma \Rightarrow F \quad \Delta \Rightarrow F \rightarrow G}{\Gamma, \Delta \Rightarrow G}$

Inference rules (ctd)

- The contradiction rule is:

  $(C)\ \frac{\Gamma \Rightarrow \bot}{\Gamma \Rightarrow F}$

- The weakening rule is:

  $(W)\ \frac{\Gamma \Rightarrow F}{\Gamma' \Rightarrow F}$ if $\Gamma \subseteq \Gamma'$

- The negation introduction and negation elimination are:

  $$\frac{\Gamma, F \Rightarrow \bot}{\Gamma \Rightarrow \neg F}$$

  $$\frac{\Gamma \Rightarrow F \quad \Delta \Rightarrow \neg F}{\Gamma, \Delta \Rightarrow \bot}$$

- The introduction and elimination rules for equivalence are:

  $$\frac{\Gamma \Rightarrow F \rightarrow G \quad \Delta \Rightarrow G \rightarrow F}{\Gamma, \Delta \Rightarrow F \leftrightarrow G}$$

  $$\frac{\Gamma \Rightarrow F \leftrightarrow G}{\Gamma \Rightarrow F \rightarrow G}$$

  $$\frac{\Gamma \Rightarrow F \leftrightarrow G}{\Gamma \Rightarrow G \rightarrow F}$$

Proof of a formula

- To prove a formula F in this system means to prove the sequent $\Rightarrow F$. Example: proof of the equivalence

  (69) $(\neg p \rightarrow p) \leftrightarrow \neg\neg p$.

  1. [axiom 67] $\neg p \rightarrow p \Rightarrow \neg p \rightarrow p$
  2. [axiom 67] $\neg p \Rightarrow \neg p$
  3. [by ($\rightarrow$ E) from 2, 1] $\neg p, \neg p \rightarrow p \Rightarrow p$
  4. [by ($\rightarrow$ E) from 3, 2] $\neg p, \neg p \rightarrow p \Rightarrow \bot$
  5. [by ($\rightarrow$ I) from 4] $\neg p \rightarrow p \Rightarrow \neg\neg p$
  6. [by ($\rightarrow$ I) from 5] $\Rightarrow (\neg p \rightarrow p) \rightarrow \neg\neg p$
  7. [axiom 67] $\neg\neg p \Rightarrow \neg\neg p$
  8. [by ($\rightarrow$ E) from 2, 7] $\neg p, \neg\neg p \Rightarrow \bot$
  9. [by (C) from 8] $\neg p, \neg\neg p \Rightarrow p$
  10. [by ($\rightarrow$ I) from 9] $\neg\neg p \Rightarrow \neg p \rightarrow p$
  11. [by ($\rightarrow$ I) from 10] $\Rightarrow \neg\neg p \rightarrow (\neg p \rightarrow p)$
  12. [by ($\wedge$ I) from 6, 11] $\Rightarrow (\neg p \rightarrow p) \leftrightarrow \neg\neg p$

Properties of the deductive system

- The deductive system is sound and complete: a formula F is provable in the system iff F is tautology.
- A formula is $intuitionistically\ provable$ if it can be proved in the deductive system without references to axiom schema (68).
- A formula F is $intuitionistically\ equivalent$ to a formula G if $F \leftrightarrow G$ is intuitionistically provable.
- Example: $\neg p \rightarrow p$ is intuitionistically equivalent to $\neg\neg p$, because the proof of (69) contains no references to the law of excluded middle.

Assertions

- According to the replacement property of intuitionistic logic, if F is a subformula of a formula F', and G' is obtained from F' by replacing an occurrence of F with another formula G, then $F' \leftrightarrow G'$ is intuitionistically derivable from $F \leftrightarrow G$.
- Example: from the fact that $\neg p \rightarrow p$ is intuitionistically equivalent to $\neg\neg p$ we can conclude that $(\neg p \rightarrow p) \wedge q$ is intuitionistically equivalent to $\neg\neg p \wedge q$.
- Every intuitionistically provable formula is valid in the logic of here-and-there.
- If two formulas are intuitionistically equivalent then they are equivalent in the logic of here-and-there.
- Assertions remain true if, instead of intuitionistic logic, we talk about the stronger deductive system, obtained from classical by replacing (68) with the axiom schema expressing $the\ weak\ law\ of\ excluded\ middle$:

  (70) $\Rightarrow \neg F \vee \neg\neg F$.

Proof of equivalence

- The formulas $p \vee \neg p$ and $\neg\neg p \rightarrow p$ are equivalent to each other in the logic of here-and-there:

1. [axiom 67] $p \vee \neg p \Rightarrow p \vee \neg p$
2. [axiom 67] $p \Rightarrow p$
3. [axiom 67] $\neg p \Rightarrow \neg p$
4. [axiom 67] $\neg\neg p \Rightarrow \neg\neg p$
5. [by ($\rightarrow$ E) from 3, 4] $\neg p, \neg\neg p \Rightarrow \bot$
6. [by (C) from 5] $\neg p, \neg\neg p \Rightarrow p$
7. [by ($\vee$ E) from 1, 2, 6] $p \vee \neg p, \neg\neg p \Rightarrow p$
8. [by ($\rightarrow$ I) from 7] $p \vee \neg p \Rightarrow \neg\neg p \rightarrow p$
9. [by ($\rightarrow$ I) from 8] $\Rightarrow (p \vee \neg p) \rightarrow (\neg\neg p \rightarrow p)$

Proof of equivalence (ctd)

10. [axiom 67] $\neg\neg p \rightarrow p \Rightarrow \neg\neg p \rightarrow p$
11. [axiom 70] $\Rightarrow \neg p \vee \neg\neg p$
12. [by ($\vee$ I) from 3] $\neg p \Rightarrow p \vee \neg p$
13. [by ($\rightarrow$ E) from 4, 10] $\neg\neg p, \neg\neg p \rightarrow p \Rightarrow p$
14. [by ($\vee$ I) from 13] $\neg\neg p, \neg\neg p \rightarrow p \Rightarrow p \vee \neg p$
15. [by ($\vee$ E) from 11, 12, 14] $\neg\neg p \rightarrow p \Rightarrow p \vee \neg p$
16. [by ($\rightarrow$ I) from 15] $\Rightarrow (\neg\neg p \rightarrow p) \rightarrow (p \vee \neg p)$
17. [by ($\wedge$ I) from 9, 16] $\Rightarrow (p \vee \neg p) \leftrightarrow (\neg\neg p \rightarrow p)$

Stronger axiom schema

- This axiom schema is stronger than (70) and can be used for establishing the validity of formulas in the logic of here-and-there as well:

  (71) $\Rightarrow F \vee (F \rightarrow G) \vee \neg G$.

- Nothing stronger would be acceptable.

- A propositional formula is valid in the logic of here-and-there iff it is provable in the deductive system obtained from intuitionistic logic by adding axiom schema (71).

- The theorem is due to Lex Hendriks.

Appendix B: Traditional definition of a Stable Model

- In [Gelfond and Lifschitz, 1988], a logic program is assumed to consist of rules of the form

  (72) $A_0 \leftarrow A_1, \ldots, A_m,\ not\ A_{m+1}, \ldots,\ not\ A_n$

  where $n \geq m \geq 0$ and $A_0, \ldots, A_n$ are atoms; we will call such expressions $traditional\ rules$.

- A finite set of traditional rules with $m = n$, that is, rules of the form

  (73) $A_0 \leftarrow A_1, \ldots, A_m$

  is essentially a Horn formula.

- The $traditional\ reduct$ of a traditional program Π relative to a set X of atoms is the set of rules (73) for all rules (72) in Π such that

$$A_{m+1}, \ldots, A_n \not\in X.$$

Traditional stable model

- According to the 1988 definition, the stable model of a traditional program Π is a set X of atoms with the following property: $X\ is\ the\ minimal\ model\ of\ the\ traditional\ reduct\ of\ \Pi\ relative\ to\ X$.

- PROPOSITION 28.
  For any traditional program Π*, a set* X *of atoms is the minimal model of the traditional reduct of* Π *relative to* X *iff* X *is a stable model of* Π*.*

Proof of Proposition 28

Let $\Pi^{\underline{X}}$ denote the traditional reduct of Π relative to X.

- Case 1: $X \not\models \Pi$. Set X is not a stable model of Π.

  On the other hand, Π contains a rule (72) such that $A_1, \ldots, A_m \in X$ and $A_0, A_{m+1}, \ldots, A_n \notin X$.

  The corresponding rule (73) in $\Pi^{\underline{X}}$ is not satisfied by X, so that X is not the minimal model of $\Pi^{\underline{X}}$.

- Case 2: $X \models \Pi$. We will show that Π^X and $\Pi^{\underline{X}}$ are satisfied by the same subsets of X.

  Since Π^X is the conjunction of the formulas R^X for all rules R of Π, and $\Pi^{\underline{X}}$ is the union of the programs $\{R\}^{\underline{X}}$ for all rules R of Π, it is sufficient to verify this claim for the case when Π is a single rule (72).

Proof of Proposition 28 (ctd)

If X contains at least one of the atoms $A_{m+1}, \ldots, A_n$, then $\Pi^{\underline{X}}$ is empty and Π^X is the tautology $\bot \rightarrow A_0^X$.

Otherwise $\Pi^{\underline{X}}$ is (73). If $A_1, \ldots, A_m \in X$, then $A_0 \in X$, because $X \models \Pi$; consequently Π^X is the result of replacing $A_{m+1}, \ldots, A_n$ in (72) with $\bot$, which is equivalent to (73).

It remains to consider the case when $A_{m+1}, \ldots, A_n \notin X$ and at least one of the atoms $A_1, \ldots, A_m$, say A_1, does not belong to X. In this case Π^X is the tautology $\bot \rightarrow A_0^X$.

On the other hand, $\Pi^{\underline{X}}$ is the rule (73) whose body contains A_1 and consequently is not satisfied by any subset of X. It follows that every subset of X satisfies $\Pi^{\underline{X}}$.

Intuitive meaning

- Rule (73) can be viewed as a rule for generating atoms: we are allowed to generate its head A_0 as soon as all atoms $A_1, \ldots A_m$ in the body have been generated.
- The minimal model of a set of rules of the form (73) is the set of all atoms that can be generated by this process, starting from the empty set.
- The traditional definition of a stable model can be thought of as an extension of this idea to rules containing negative literals in the body.
- A rule (72) allows us to generate A_0 as soon as we generated the atoms $A_1, \ldots, A_m$ $provided\ that\ non\ of\ the\ atoms$ $A_{m+1}, \ldots, A_n$ $can\ be\ generated\ using\ the\ rules\ of\ the\ program.$
- There is a vicious circle in this sentence: to decide whether a rule of Π can be used to generate a new atom, we need to know which atoms can be generated using the rules of Π.
- The traditional definition of a stable model overcomes this difficulty using a $fixpoint\ construction.$

Fixpoint construction

- Take a set X that you suspect may be exactly the set of atoms that can be generated using the rules of Π.
- Under this assumption, Π has the same meaning as the traditional reduct of Π relative to X, which is a set of rules of the form (73).
- Consider the minimal model of the traditional reduct.
- If this model is exactly identical to the set X that we started with, then X was a $good\ guess$; it is indeed a stable model of Π.

Relation with Prolog

- The definition of a stable model for traditional programs can be viewed as a possible definition of a $correct$ answer to a query in Prolog.
- Let Π be a Prolog program without variables: the set of ground rules obtained from a Prolog program with variables by replacing each rule with all its ground instances.
- If Π is a traditional program with a unique stable model then the correct answer to a ground query A is yes or no depending on whether A belongs to that model.
- From this perspective, a program with several stable models is bad: it does not provide an unambiguous specificaiton for the behavior of a Prolog system. Programs without answer sets are bad also.
- In ASP, on the other hand, programs without a unique answer set are quite useful: they correspond to computational problems with many solutions, or with no solutions.
- The concept of stable model is only one of several available definitions of the semantics of negation as failure. Two other definitions are based on program completion and the well-founded model.
- These three definitions are not equivalent to each other, but each provides an adequate description of the behavior of Prolog.