Nonmonotonic logics—recent advances

Mirosław Truszczyński

Department of Computer Science
University of Kentucky

August 3-7, 2008

Introduction

McCarthy and Hayes on AI, 1969

> [...] intelligence has two parts,

which we shall call the epistemological and the heuristic.
The epistemological part is the representation of the world in such a form that the solution of problems follows from the facts expressed in the representation. The heuristic part is the mechanism that on the basis of the information solves the problem and decides what to do.

McCarthy and Hayes on AI, 1969

Knowledge representation and reasoning (KRR) — core of AI

- Epistemological part $\longrightarrow$ modeling
- Knowledge representation
- Heuristic part $\longrightarrow$ search for models or proofs
- Reasoning
- There is more to AI now – but KRR remains its core
- How to approach it?
  - Use classical logic — it is "descriptively universal" and reasoning can be automated
    Early proposal of McCarthy

The goal *(expanding and rephrasing Gelfond and Leone, 2002)*

- To design and study languages to capture knowledge about: environments, their entities and their behaviors
- To develop tools to support of reasoning from theories in these languages

Challenges

With FOL – things are not so easy

- Incomplete information

  (new information may invalidate earlier inferences – defeasible reasoning)
- Qualification problem

  (we do not check for potato in tailpipe before starting the engine)
- Ramification problem

  (describing side effects)
- Frame problem

  (moving an object does not change its color)
- Rules with exceptions (defaults)
- Definitions – most notably *inductive* definitions

A typical scenario

Where to find Professor Jones?

Professor Jones likes to have a good espresso after lunch in a campus café. You need to talk to her about a grant proposal. It is about 1:00pm and, under normal circumstances, Professor Jones sticks to her daily routine.
Thus, you draw a plausible conclusion that she is presently enjoying her favorite drink. You decide to go to the café and meet her there. As you get near the student center, where the cafe is located, you see people streaming out of the building. One of them tells you about the fire alarm that just went off.

The new piece of information invalidates the normality assumption and so the conclusion about the present location of Professor Jones, too.

Key questions

- How to model such knowledge?
- How to reason with the representation?

A typical scenario

Where to find Professor Jones?

Professor Jones likes to have a good espresso after lunch in a campus café. You need to talk to her about a grant proposal. It is about 1:00pm and, under normal circumstances, Professor Jones sticks to her daily routine.
Thus, you draw a plausible conclusion that she is presently enjoying her favorite drink. You decide to go to the café and meet her there. As you get near the student center, where the cafe is located, you see people streaming out of the building. One of them tells you about the fire alarm that just went off.

The new piece of information invalidates the normality assumption and so the conclusion about the present location of Professor Jones, too.

Key questions

- How to model such knowledge?
- How to reason with the representation?

Normative statements, defaults

Professors normally teach (or, professors teach by *default*)

- How to formalize this statement?
- $p \rightarrow t$

  Not quite right, normality not reflected
- $p \wedge \neg ab \rightarrow t$?

  Not quite right, abnormality must be established

Normative statements, defaults

Indeed ...

- $p \wedge \neg ab \supset t$
- Given p, can we derive t?
- No - need to know about normality/abnormality of p
  - $\{p, t\}$, $\{p, t, ab\}$ and $\{p, ab\}$ are models
  - cannot infer t
- Thus, classical logics are not best equipped for the task!!

Change of semantics needed

- For instance: minimal models wrt ab
  - $\{p, t\}$ — the only model left
  - t follows!!
- Circumscription by McCarthy

Normative statements, defaults

Indeed ...

- $p \wedge \neg ab \supset t$
- Given p, can we derive t?
- No - need to know about normality/abnormality of p
  - $\{p, t\}$, $\{p, t, ab\}$ and $\{p, ab\}$ are models
  - cannot infer t
- Thus, classical logics are not best equipped for the task!!

Change of semantics needed

- For instance: minimal models wrt ab
  - $\{p, t\}$ — the only model left
  - t follows!!
- Circumscription by McCarthy

Planning

How to move blocks around?

- Find a plan!
- Blocks on the table are arranged in stacks
- They need to be rearranged into different stacks
- At each step
  - only top block in each stack can be moved
  - can be placed on top of another stack or on the table
- How to describe prerequisites for actions and their effects?

  qualification problem, ramification problem
- How to describe what changes when actions are executed and what does not!!

  frame problem

Planning

Games and puzzles

- Missionaries and cannibals
- 15-puzzle

Constraints

How to find a good route?

- Given:
  - a set of places to visit
  - the set of pairs of places that have direct connections
- Find a closed route that takes you through each place exactly once
- Hamiltonian cycle problem
- Traveling salesman problem
- Defining the concept of reachability is hard!

Constraints

More generally, combinatorial search problems

- Propositional satisfiability
- Graph problems (coloring, independent sets, cliques)
- Combinatorial optimization (largest size clique)
- Sudoku

Back to “Professors normally teach”

Another formalism: LP with stable models

- $t \leftarrow p, \mathit{not}\ ab$
- Together with p, there is one stable model: $\{p, t\}$
- Stable logic programming (answer-set programming) by Gelfond and Lifschitz

Nonmonotonicity

In each case, learning *ab defeats* earlier conclusion!

- $\{p \wedge \neg ab \supset t, \quad p, \quad ab\}$
- One minimal model: $\{p, ab\}$
- $\{t \leftarrow p, not\ ab. \quad p. \quad ab.\}$
- One stable model: $\{p, ab\}$

Not what happens in classical logics where ...

- If $T \models \varphi$ and $T \subseteq T'$, then $T' \models \varphi$
- Monotonicity of classical logics
- Undesirable aspect of classical logics (from KR perspective)
- Difficulty in modeling incomplete information and defeasible reasoning

Nonmonotonicity

In each case, learning *ab defeats* earlier conclusion!

- $\{p \wedge \neg ab \supset t, \quad p, \quad ab\}$
- One minimal model: $\{p, ab\}$
- $\{t \leftarrow p, not\ ab. \quad p. \quad ab.\}$
- One stable model: $\{p, ab\}$

Not what happens in classical logics where ...

- If $T \models \varphi$ and $T \subseteq T'$, then $T' \models \varphi$
- Monotonicity of classical logics
- Undesirable aspect of classical logics (from KR perspective)
- Difficulty in modeling incomplete information and defeasible reasoning

Non-monotonic logics

Proposed in response to challenges of KRR

- Language of logic with non-classical semantics
- Model preference
  - circumscription *(McCarthy 1977)*
- Fixpoint conditions defining belief sets
  - default logic *(Reiter 1980)*
  - autoepistemic logic *(Moore 1984)*
  - logic programming with stable-model semantics (more manageable fragment of default logic) *(Gelfond-Lifschitz, 1988)*
  - ID-logic *(Denecker 1998, 2000; Denecker-Ternovska 2004)*
- Emphasize both modeling and reasoning

Introduction to default logic

Introduction to default logic

Default

- $d = \frac{\varphi : \psi_1, \ldots, \psi_m}{\vartheta}$
  - φ — premise; notation: $p(d)$
  - ϑ — consequent; notation: $c(d)$
  - $\psi_1, \ldots, \psi_m$ — justifications; notation: $j(d) = \{\psi_1, \ldots, \psi_m\}$
- Interpretation
  - if φ has been derived and all ψ_i are consistent, conclude ϑ
  - like an inference rule $\frac{\varphi}{\vartheta}$ modulo exceptions $\neg\psi_i$
- $[\frac{\varphi : \psi_1, \ldots, \psi_m}{\vartheta}] := \frac{\varphi}{\vartheta}$
- $[d]$ and $[D]$ (for a set of defaults)

Example

Defaults

- $$\frac{p : \neg q}{r} \qquad \frac{p : \neg r, \neg s}{q} \qquad \frac{r : \neg s}{s}$$
- The monotone rules ($d \rightarrow [d]$)

$$\frac{p}{r} \qquad \frac{p}{q} \qquad \frac{r}{s}$$

- Exceptions

$$q \qquad r, s \qquad s$$

Introduction to default logic

Default theory

- A pair of sets (D, W):
  - D — a set of defaults
  - W — a set of initial assumptions
- Semantics! What does a default theory entail?

Example

Default theory

- $W = \{p\}$

  $D = \left\{ \frac{p : \neg q}{r}, \quad \frac{p : \neg r, \neg s}{q}, \quad \frac{r : \neg s}{s} \right\}$

Defaults in a context

$\varphi : \psi_1, \ldots, \psi_m / \vartheta$ is S-enabled if

- $S \nvdash \neg\psi_i$, $1 \leq i \leq m$
- D_S
- $[D_S]$ — for the corresponding monotone rules

$\varphi : \psi_1, \ldots, \psi_m / \vartheta$ is S-applicable if

- $S \nvdash \neg\psi_i$, $1 \leq i \leq m$
- $S \vdash \varphi$
- $D(S)$
- (S-generating)

Defaults in a context

$\varphi : \psi_1, \ldots, \psi_m / \vartheta$ is S-enabled if

- $S \nvdash \neg\psi_i$, $1 \leq i \leq m$
- D_S
- $[D_S]$ — for the corresponding monotone rules

$\varphi : \psi_1, \ldots, \psi_m / \vartheta$ is S-applicable if

- $S \nvdash \neg\psi_i$, $1 \leq i \leq m$
- $S \vdash \varphi$
- $D(S)$
- (S-generating)

Reasoning with default theories

Make assumptions

- Select a context
- Justify the selection

How?

- Use consequents of S-applicable defaults together with W to derive S by means of propositional logic
- Use "monotone" parts of S-enabled defaults and propositional logic to derive S from W

Reasoning with default theories

Make assumptions

- Select a context
- Justify the selection

How?

- Use consequents of S-applicable defaults together with W to derive S by means of propositional logic
- Use “monotone” parts of S-enabled defaults and propositional logic to derive S from W

Weak extensions (expansions) of (D, W)

Use S-applicable defaults

- Formally:

$$S = Cn(W \cup c(D(S)))$$

- Circular justifications
  - $W = \emptyset$
  - $D = \{\frac{p : q}{p}\}$
  - $S_1 = Cn(\emptyset)$; $D(S_1) = \emptyset$; all checks: $c(D(S_1)) = \emptyset$
  - $S_2 = Cn(p)$; $D(S_2) = D$; all checks: $c(D(S_2)) = \{p\}$

Extensions of (D, W)

Use S-enabled defaults

- Formally:

$$S = Cn^{[D_S]}(W)$$

- No circularity
  - $W = \emptyset$
  - $D = \{\frac{p : q}{p}\}$
  - $S_1 = Cn(\emptyset)$; $D_{S_1} = \emptyset$; all checks
  - $S_2 = Cn(p)$; $D_{S_2} = \{\frac{p : q}{p}\}$; $[D_{S_2}] = \{\frac{p}{p}\}$; does not work
    the rule $\frac{p}{p}$ will never be applied

Extensions of (D, W)

A larger example

- $W = \{p\}$

  $D = \left\{ \frac{p:\neg q}{r}, \quad \frac{p:\neg r, \neg s}{q}, \quad \frac{r:\neg s}{s} \right\}$
- Exactly one extension: $S = Cn(p, q)$
- Default proofs w.r.t. S use:

$$p, \quad \frac{p}{q} \quad \text{and} \quad \frac{r}{s}$$

- Exactly the consequences of p and q can be derived!

Extensions of (D, W)

A larger example

- $W = \{p\}$

  $D = \left\{ \frac{p:\neg q}{r}, \quad \frac{p:\neg r, \neg s}{q}, \quad \frac{r:\neg s}{s} \right\}$
- $S = Cn(p, r)$ is not an extension
- Default proofs w.r.t. S use

$$p, \quad \frac{p}{r} \quad \text{and} \quad \frac{r}{s}$$

- Can prove s even though not assumed!

Extensions of (D, W)

Reiter's original definition

- Let (D, W) be a default theory. For every set S, there is a least set U such that:
  - $W \subseteq U$
  - $Cn(U) = U$
  - Whenever $\frac{\varphi : \psi_1, \ldots, \psi_m}{\vartheta}$ is a default rule in D, $\varphi \in U$ and $\neg\psi_1, \ldots, \neg\psi_m \notin Cn(S)$ then $\vartheta \in U$
- Denote this set by $\Gamma_{(D,W)}(S)$
- (Reiter, 1980) S is an extension of (D, W) if

$$S = \Gamma_{(D,W)}(S)$$

- Connection: $\Gamma_{(D,W)}(S) = Cn^{[D_S]}(W)$

An application

Modeling inertia (change requires an action)

- Modeling inertia (change requires an action)

$$\frac{in\_location(X, L, T) : \neg abnormal(X, T)}{in\_location(X, L, T+1)}$$

$$move(X, L1, L2, T) \supset in\_location(X, L2, T+1)$$

$$move(X, L1, L2, T) \supset abnormal(X, T)$$

Default Logic

Properties

- If S is an extension of (D, W), then

$$S = Cn(W \cup c(D(S)))$$

- Weak extensions are extensions
- $Cn^{[D_{(\cdot)}]}(W)$ and $\Gamma_{(D,W)}(\cdot)$ are antimonotone
- Fixpoints of $Cn^{[D_{(\cdot)}]}$ and $\Gamma_{(D,W)}(\cdot)$ form an antichain
- Extensions of a default theory form an antichain

Normal default logic

Normal defaults and normal default theories

- A normal default is a default of the form:

$$\frac{\varphi : \psi}{\psi}$$

- (D, W) is normal if all defaults in D are normal
- A normal default theory always has an extension
- If (D, W) is normal and W is consistent, then all extensions of (D, W) are consistent
- If (D, W) is a normal default theory and T_1 and T_2 are *distinct* extensions of (D, W), then $T_1 \cup T_2$ is inconsistent
- Let D_1 and D_2 be sets of normal defaults. If T is an extension of (D_1, W) then there is T such that T is an extension of $(D_1 \cup D_2, W)$ and $T_1 \subseteq T$

  Semimonotonicity

Simple application

Closed World Assumption

- Let W be a propositional theory. Define:

$$CWA(W) = Cn(W \cup \{\neg p : p \text{ is an atom and } W \nvdash p\})$$

- W is *CWA*-consistent if $CWA(W)$ is consistent
- $CWA(W)$ is complete
- $CWA(W)$ may be inconsistent (consider $W = \{a \vee b\}$)
- If W is a consistent Horn theory then W is *CWA*-consistent. Moreover, $CWA(W)$ is precisely the theory of the least (w/r to inclusion) model of W

CWA, cont'd

CWA and NDL

- Define:

$$D^{cwa} = \left\{ \frac{: \neg p}{\neg p} : p \text{ — an atom} \right\}$$

- W is *CWA*-consistent if and only if
  - W is consistent, and
  - (D^{cwa}, W) has a unique extension

Reasoning tasks

Given a default theory (D, W) and a formula φ (in the last three)

EXISTENCE	Decide whether (D, W) has an extension
IN-SOME	Decide whether φ is in some extension for (D, W) (*credulous reasoner model*)
NOT-IN-ALL	Decide whether there is an extension of (D, W) not containing φ
IN-ALL	Decide whether φ is in all extensions of (D, W) (*skeptical reasoner model*)

Complexity

Results

- EXISTENCE, IN-SOME and NOT-IN-ALL are Σ_2^P-complete
- IN-ALL is Π_2^P-complete
- For normal default theories, IN-SOME and NOT-IN-ALL are Σ_2^P-complete, and IN-ALL is Π_2^P-complete
- The problem of deciding whether a finite default theory (D, W) possesses at least one consistent extension is Σ_2^P-complete

Importance of default logic

Generality and change of perspective

- Generalizes Closed World Assumption
- Generalizes logic programming with stable model semantics
- Captures propositional circumscription
- Captures an important fragment of autoepistemic logic
- Changes the way logic is used in KRR
  - extensions as opposed to proofs
  - answer-set programming

Logic programing as answer-set programming

Some logic terminology

Language

- Constant, variable, function and predicate symbols
- Terms: strings built recursively from constant, variable and function symbols
- $c, X, f(c, X), f(f(c, X), f(X, f(X, c)))$
- Atoms: built of predicate symbols and terms
- $p(X, c, f(a, Y))$

Horn logic programming

Horn clause

- $p \leftarrow q_1, \ldots, q_k$
  - where p, q_i are atoms
- Clauses are universally quantified
  - special sentences
- Intuitive reading: if $q_1, \ldots, q_k$ then p

Horn program

- A collection of Horn clauses

Horn logic programming

Horn clause

- $p \leftarrow q_1, \ldots, q_k$
  - where p, q_i are atoms
- Clauses are universally quantified
  - special sentences
- Intuitive reading: if $q_1, \ldots, q_k$ then p

Horn program

- A collection of Horn clauses

More terminology

Herbrand model

- Ground terms: no variable symbols
- Herbrand universe: collection of all ground terms
- Ground atoms: atoms built of predicate symbols and ground terms
- $p(a, c, f(a, a))$
- Herbrand base: collection of all ground atoms
- Herbrand model: subset of an Herbrand base

Horn logic programming

Semantics

- Given by Herbrand models
  - $grnd(P)$: the set of all ground instances of clauses in P
  - Thus, no difference between P and $grnd(P)$
- Key question: which ground facts hold in every Herbrand model of P?
- Sufficient to restrict to Herbrand models contained in $HB(P)$
  - Herbrand universe of P, $HU(P)$
  - Herbrand base of P, $HB(P)$
  - Ground atoms not in $HB(P)$ are not true in all Herbrand models

We can say more

Least Herbrand model

- Every Horn program P has a least Herbrand model $LM(P)$
  - the intersection of a set of Herbrand models of a Horn program is a Herbrand model of the program
  - $HB(P)$ is an Herbrand model of P
  - the intersection of all models is a least Herbrand model (and it is containmed in $HB(P)$)
- Single intended Herbrand model
- For a ground t, $P \models p(t)$ if and only if $p(t) \in LM(P)$
- For ground t, if $P \not\models p(t)$, defeasibly conclude $\neg p(t)$
- Closed World Assumption (CWA)

Computing with Horn programs

What do they specify, what can they express?

- A Horn program P specifies a subset X of the Herbrand universe for P, $HU(P)$, if for some predicate symbol p occurring in P we have:

$$X = \{t \in HU(P) : p(t) \in LM(P)\}$$

- Finite Horn programs specify precisely the r.e. sets and relations

  Smullyan, 1968, Andreka and Nemeti, 1978

Computing with Horn programs

Possible issues?

- Function symbols necessary!
- List constructor $[\cdot|\cdot]$ used to define higher-order objects
- Terms - basic data structures
- Queries (goals):
  - ?p(t) - is $p(t)$ entailed?
  - ?p(X) - for what ground t, is $p(t)$ entailed?
- Proofs provide answers
- SLD-resolution
- Unification - basic mechanism to manipulate data structures
- Extensive use of recursion
- Leads to Prolog

Example

Manipulating lists: *append* and *reverse*

$$append([\,], X, X).$$
$$append([X|Y], Z, [X|T]) \leftarrow append(Y, Z, T).$$

$$reverse([\,], [\,]).$$
$$reverse([X|Y], Z) \leftarrow append(U, [X], Z), reverse(Y, U).$$

- both relations defined recursively
- terms represent complex objects: lists, sets, ...

Example, cont'd

Playing with *reverse*

- Problem: reverse list $[a, b, c]$
  - Ask query $? - reverse([a, b, c], X)$.
  - A proof of the query yields a substitution: $X = [c, b, a]$
  - The substitution constitutes an answer
- Query $? - reverse([a|X], [b, c, d, a])$ returns $X = [d, c, b]$
- Query $? - reverse([a|X], [b, c, d, b])$ returns no substitutions (there is no answer)

Example, cont'd

Observations

- Techniques to search for proofs — the key
- Understanding of the resolution mechanism is important
- It may make a difference which logically equivalent form is used:
  - $reverse([X|Y], Z) \leftarrow append(U, [X], Z), reverse(Y, U).$
  - $reverse([X|Y], Z) \leftarrow reverse(Y, U), append(U, [X], Z).$
  - termination vs. non-termination for query:
    $? - reverse([a|X], [b, c, d, b])$
- Is it truly knowledge representation?
  - is it truly declarative?
  - implementations are not!
- Nonmonotonicity quite restricted

Negation in the body

Why negation?

- Natural linguistic concept
- Facilitates knowledge representation (declarative descriptions and definitions
- Needed for modeling convenience
- Clauses of the form:

$$p(\vec{X}) \leftarrow q_1(\vec{X_1}), \ldots, q_k(\vec{X_k}), not\ r_1(\vec{Y_1}), \ldots, not\ r_l(\vec{Y_l})$$

- Things get more complex!

Semantics of programs with negation

Observations

- Still Herbrand models
- Still restricted to $HB(P)$
- But — usually no least Herbrand model!
- Program

  $$a \leftarrow not\ b$$
  $$b \leftarrow not\ a$$

  has two minimal Herbrand models: $M_1 = \{a\}$ and $M_2 = \{b\}$.
- Identifying a single intended model a major issue

Semantics of programs with negation

Great Logic Programming Schism

- Single intended model approach
  - continue along the lines of Prolog
- Multiple intended model approach
  - branch into answer-set programming

Single intended model approach

"Better" Prolog

- Extensions of Horn logic programming
  - Perfect semantics of stratified programs
  - 3-val well-founded semantics for (arbitrary) programs
- Top-down computing based on unification and resolution
- XSB – David Warren at SUNY Stony Brook
- Will come back to it

Multiple intended models

Answer-set programming

- Semantics assigns to a program not one but many intended models!
  - for instance, all stable or supported models (to be introduced soon)
- How to interpret these semantics?
  - skeptical reasoning: a ground atom is cautiously entailed if it belongs to all intended models
  - intended models represent different possible states of the world, belief sets, solutions to a problem
- Nonmonotonicity shows itself in an essential way
  - as in default logic

Normal logic programming

Preliminary observations and comments

- Logic programs with negation
- Still interested only in Herbrand models
- Thus, enough to consider propositional case
- Supported model semantics
- Stable model semantics
- Connection to propositional logic (Clark's completion, tightness, loop formulas)
- Kripke-Kleene and well-founded semantics
- Strong and uniform equivalence

Normal logic programming — propositional case

Syntax

- Propositional language over a set of atoms At (possibly infinite)
- Clause r

$$a \leftarrow b_1, \ldots, b_m, not\ c_1, \ldots, not\ c_n$$

  - a, b_i, c_j are atoms
  - a is the head of the clause: $hd(r)$
  - literals b_i, $not\ c_j$ form the body of the rule: $bd(r)$
  - $\{b_1, \ldots, b_m\}$ - positive body $bd^+(r)$
  - $\{c_1, \ldots, c_n\}$ - negative body $bd^-(r)$

One-step provability operator

Basic tool in LP

van Emden, Kowalski 1976

- Operator on interpretations (sets of atoms)
- $T_P(I) = \{hd(r) : I \models bd(r)\}$
- If P is Horn, T_P is monotone
  - Let $I \subseteq J$
  - Let $bd(r) = b_1, \ldots, b_m$ (no negation!)
  - If $I \models bd(r)$ than $J \models bd(r)$
  - Thus, $T_P(I) \subseteq T_P(J)$
  - Least fixpoint of T_P exists and coincides with the least Herbrand model of P
- In general, not the case (due to negation)
  - $\emptyset \models not\ a$
  - but $\{a\} \not\models not\ a$

Supported-model semantics

Definition and some observations

- $M \subseteq At$ is a supported model of P if $T_P(M) = M$
- Supported models are indeed models
  - let $M \models bd(r)$
  - then $hd(r) \in T_P(M)$
  - and so, $hd(r) \in M$
- Supported models are subsets of $At(P)$ (the Herbrand base of P)
- Thus, they are Herbrand models

Supported models - example

Program $p \leftarrow \mathit{not}\ q$

- One supported model: $M_1 = \{p\}$
- $M_2 = \{q\}$ - not supported (but model)
- p "follows"
- If q included in the program (more exactly: a rule $q \leftarrow$)
  - Just one supported model: $M_1 = \{q\}$.
  - p does not 'follow"
  - nonmonotonicity

Supported models - example

Program $p \leftarrow p$

- Two supported models: $M_1 = \emptyset$ and $M_2 = \{p\}$
- The second one is self-supported (circular justification)
- A problem for KR

Clark's completion

Rules as implications

- $bd^{\wedge}(r)$ the conjunction of all literals in the body of r

  with all not c replaced with $\neg c$
- $cmpl^{\leftarrow}(P) = \{bd^{\wedge}(r) \rightarrow hd(r) : r \in P\}$

Clark's completion

Rules as definitions

- Notation: $def_P(a) = \bigvee\{bd^{\wedge}(r) : hd(r) = a\}$
- Note: if a not the head of any rule in P, $def_P(a) = \bot$
- $cmpl^{\rightarrow}(P) = \{a \rightarrow def_P(a) : a \in At\}$
- $cmpl(P) = cmpl^{\leftarrow}(P) \cup cmpl^{\rightarrow}(P)$
- Note: if $a \notin At(P)$, $cmpl(P) \models \neg a$

Clark's completion

Connection to supported models

- A set $M \subseteq At$ is a supported model of a program P if and only if M is a model (in a standard sense) of $cmpl(P)$
- Connection to SAT
- Allows us to use SAT solvers to compute supported models of P

Stable model semantics

Supported models of interest, but ...

- Some supported models based on circular arguments
- Some more stringent bases for selecting intended models needed

Stable model semantics

Gelfond-Lifschitz reduct

- P — logic program
- M — set of atoms
- Reduct P^M
  - for each $a \in M$ remove rules with *not a* in the body
  - remove literals *not a* from all other rules

Stable model semantics

Definition through reduct

- Reduct P^M is a Horn program
- It has the least model $LM(P^M)$
- M is a stable model of P if

$$M = LM(P^M)$$

Stable model semantics

And now through Gelfond-Lifschitz operator

- $GL_P(M) = LM(P^M)$
- M is a stable model if and only if

$$M = GL_P(M)$$

- GL_P is antimonotone
- For $M \subseteq N$:

$$GL_P(N) \subseteq GL_P(M)$$

Stable models — examples

Multiple stable models

$$
\begin{array}{lll}
p & \leftarrow & q, \mathit{not}\ s \\
r & \leftarrow & p, \mathit{not}\ q, \mathit{not}\ s \\
s & \leftarrow & \mathit{not}\ q \\
q & \leftarrow & \mathit{not}\ s
\end{array}
$$

- Two stable models: $M_1 = \{p, q\}$ and $M_2 = \{s\}$

No stable models

$$
p \leftarrow \mathit{not}\ p
$$

- No stable models!!

Stable models — examples

Multiple stable models

$$
\begin{aligned}
p &\leftarrow q, \mathit{not}\ s \\
r &\leftarrow p, \mathit{not}\ q, \mathit{not}\ s \\
s &\leftarrow \mathit{not}\ q \\
q &\leftarrow \mathit{not}\ s
\end{aligned}
$$

- Two stable models: $M_1 = \{p, q\}$ and $M_2 = \{s\}$

No stable models

$$p \leftarrow \mathit{not}\ p$$

- No stable models!!

Stable models — properties

Stable models are models!

- Let M be a stable model
- M is a model of all rules that are removed from the program when forming the reduct
- M is a model of every rule that contributes to the reduct
- Indeed, each such rule is subsumed by a rule in the reduct and M satisfies all rules in the reduct

Stable models — properties

Stable models are minimal models!

- Let N be a stable model and M a model s.t. $M \subseteq N$
- Then

$$N = GL_P(N) \subseteq GL_P(M) \subseteq M$$

- Thus, $N \subseteq M$ and so $N = M$
- The minimality of N follows
- Stable models form an antichain!

Stable models — properties

Lemma: If M model of P, $GL_P(M) \subseteq M$

- Since M model of P, then M is a model of P^M
- $a \leftarrow b_1, \ldots, b_m$ - a rule in reduct
- $a \leftarrow b_1, \ldots, b_m, not\ c_1, \ldots, not\ c_n$ - the original rule in P
- M satisfies the latter, and it satisfies every $not\ c_i$ (as $c_i \notin M$)
- Thus, M satisfies the reduct rule

Stable models — properties

Connection to supported models

- If M is a stable model of P then it is a supported model of P
- Let M be a stable model of P
- Then M model of P and so, $T_P(M) \subseteq M$
- $r = \quad a \leftarrow b_1, \ldots, b_m, not\ c_1, \ldots, not\ c_n$ - a rule in P such that $M \models bd(r)$
- Then $r' \quad = a \leftarrow b_1, \ldots, b_m$ belongs to the reduct P^M
- And $M \models bd(r')$
- Since M is a model of P^M, $a \in M$
- Hence, $T_P(M) \subseteq M$ and so, $M = T_P(M)$
- That is, M is supported!!

Fages Lemma

Fages 1994

But ...

- The converse not true, in general (as it should not be)

Counterexample

- $p \leftarrow p$
- $\{p\}$ is supported but not stable
- Positive dependency of p on itself is a problem

Fages Lemma

Fages 1994

But ...

- The converse not true, in general (as it should not be)

Counterexample

- $p \leftarrow p$
- $\{p\}$ is supported but not stable
- Positive dependency of p on itself is a problem

Fages Lemma

Positive dependency graph $G^+(P)$

- Atoms of P are vertices
- (a, b) is an edge in $G^+(P)$ if for some $r \in P$: $\quad hd(r) = a$, $b \in bd^+(r)$

Tight programs

- P is tight if $G^+(P)$ is acyclic
- Alternatively, if there is a labeling of atoms with non-negative integers $(a \mapsto \lambda(a))$ s.t.
- for every rule $r \in P$

$$\lambda(hd(r)) > \max\{\lambda(b) : b \in bd^+(r)\}$$

- Connection to topological ordering of positive dependency graphs

Fages Lemma

Positive dependency graph $G^+(P)$

- Atoms of P are vertices
- (a, b) is an edge in $G^+(P)$ if for some $r \in P$: $\quad hd(r) = a$, $b \in bd^+(r)$

Tight programs

- P is tight if $G^+(P)$ is acyclic
- Alternatively, if there is a labeling of atoms with non-negative integers $(a \mapsto \lambda(a))$ s.t.
- for every rule $r \in P$

$$\lambda(hd(r)) > \max\{\lambda(b) : b \in bd^+(r)\}$$

- Connection to topological ordering of positive dependency graphs

Fages Lemma

The statement — finally

- If P is tight then every supported model is stable
- Intuitively, circular support not possible

Fages Lemma — example

Program P

$$
\begin{array}{lll}
p & \leftarrow & q, not\ s \\
r & \leftarrow & p, not\ q, not\ s \\
s & \leftarrow & not\ q \\
q & \leftarrow & not\ s
\end{array}
$$

Graph $G^+(P)$

P is tight

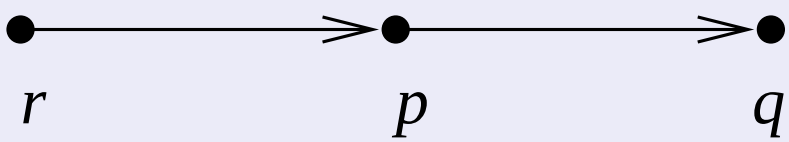
- $\{p, q\}$ and $\{s\}$ are supported models of P
  - $T_P(\{p, q\}) = \{p, q\}$
  - $T_P(\{s\}) = \{s\}$
- Thus, they are stable (which we verified directly earlier)

Fages Lemma — example

Program P

$$
\begin{array}{lll}
p & \leftarrow & q, not\ s \\
r & \leftarrow & p, not\ q, not\ s \\
s & \leftarrow & not\ q \\
q & \leftarrow & not\ s
\end{array}
$$

Graph $G^+(P)$



P is tight

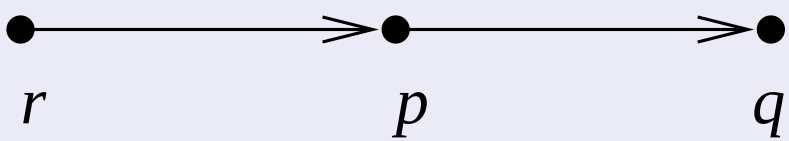
- $\{p, q\}$ and $\{s\}$ are supported models of P
  - $T_P(\{p, q\}) = \{p, q\}$
  - $T_P(\{s\}) = \{s\}$
- Thus, they are stable (which we verified directly earlier)

Fages Lemma — example

Program P

$$
\begin{array}{lll}
p & \leftarrow & q, not\ s \\
r & \leftarrow & p, not\ q, not\ s \\
s & \leftarrow & not\ q \\
q & \leftarrow & not\ s
\end{array}
$$

Graph $G^+(P)$



P is tight

- $\{p, q\}$ and $\{s\}$ are supported models of P
  - $T_P(\{p, q\}) = \{p, q\}$
  - $T_P(\{s\}) = \{s\}$
- Thus, they are stable (which we verified directly earlier)

Fages Lemma

Proof

- Let P be tight and M be its supported model
- Then M is a model of P^M
- Let N be a model of P^M
- Claim: for every k, if $a \in M$ and $\lambda(a) < k$, then $a \in N$
- Holds for $k = 0$ (trivially)
- Let $a \in M$ and $\lambda(a) = k$
- Since M supported, there is $r \in P$ such that $a = hd(r)$ and $M \models bd(r)$
- In particular, $bd^+(r) \subseteq M$ and so, $bd^+(r) \subseteq N$ (by I.H.)
- Since $M \models bd(r)$, M contributes to the reduct
- Since N is a model of P^M, $a \in N$
- It follows that $M = LM(P^M)$

A generalization

Erdem and Lifschitz, 2000

Relativized tightness

- Let $X \subseteq At(P)$
- P is tight on X if the program consisting of rules r such that $bd^+(r) \subseteq X$ is tight

Generalization

- If P is tight on X and M is a supported model of P such that $M \subseteq X$, then M is stable

A generalization

Erdem and Lifschitz, 2000

Relativized tightness

- Let $X \subseteq At(P)$
- P is tight on X if the program consisting of rules r such that $bd^+(r) \subseteq X$ is tight

Generalization

- If P is tight on X and M is a supported model of P such that $M \subseteq X$, then M is stable

Generalized Fages Lemma — example

Program P

$$
\begin{array}{lll}
p & \leftarrow & q, not\ s \\
r & \leftarrow & p, not\ q, not\ s \\
s & \leftarrow & not\ q \\
q & \leftarrow & not\ s \\
p & \leftarrow & r
\end{array}
$$

Graph $G^+(P)$

P is not tight

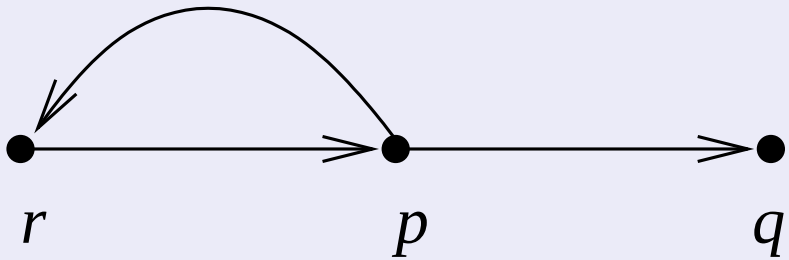
- $\{p, q\}$ and $\{s\}$ are still supported models of P
  - $T_P(\{p, q\}) = \{p, q\}$
  - $T_P(\{s\}) = \{s\}$
- Since P is tight on each of them, they are stable

Generalized Fages Lemma — example

Program P

$$
\begin{array}{lll}
p & \leftarrow & q, not\ s \\
r & \leftarrow & p, not\ q, not\ s \\
s & \leftarrow & not\ q \\
q & \leftarrow & not\ s \\
p & \leftarrow & r
\end{array}
$$

Graph $G^+(P)$



P is not tight

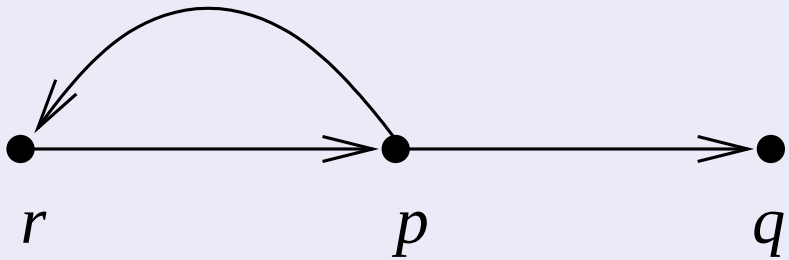
- $\{p, q\}$ and $\{s\}$ are still supported models of P
  - $T_P(\{p, q\}) = \{p, q\}$
  - $T_P(\{s\}) = \{s\}$
- Since P is tight on each of them, they are stable

Generalized Fages Lemma — example

Program P

$$
\begin{array}{lll}
p & \leftarrow & q, not\ s \\
r & \leftarrow & p, not\ q, not\ s \\
s & \leftarrow & not\ q \\
q & \leftarrow & not\ s \\
p & \leftarrow & r
\end{array}
$$

Graph $G^+(P)$



P is not tight

- $\{p, q\}$ and $\{s\}$ are still supported models of P
  - $T_P(\{p, q\}) = \{p, q\}$
  - $T_P(\{s\}) = \{s\}$
- Since P is tight on each of them, they are stable

Loops and loop formulas

Lin and Zhao, 2002

External support formula for $Y \subseteq At(P)$

- For a rule r:
- $bd^{\wedge}(r)$ the conjunction of all literals in the body of r

  with all not c replaced with $\neg c$
- $ES_P(Y)$ the disjunction of $bd^{\wedge}(r)$ for all $r \in P$ st:
  - $hd(r) \in Y$
  - $bd^{+}(r) \cap Y = \emptyset$

Observations

- $ES_P(\emptyset) = \top$
- $ES_P(\{a\}) = def_P(a)$

  cf. Clark's completion

Loops and loop formulas

Lin and Zhao, 2002

External support formula for $Y \subseteq At(P)$

- For a rule r:
- $bd^{\wedge}(r)$ the conjunction of all literals in the body of r

  with all not c replaced with $\neg c$
- $ES_P(Y)$ the disjunction of $bd^{\wedge}(r)$ for all $r \in P$ st:
  - $hd(r) \in Y$
  - $bd^{+}(r) \cap Y = \emptyset$

Observations

- $ES_P(\emptyset) = \top$
- $ES_P(\{a\}) = def_P(a)$

  cf. Clark's completion

A characterization of stable models

The following conditions are equivalent

- X is a stable model of P
- X is a model of $cmpl^{\leftarrow}(P) \cup \{Y^{\wedge} \rightarrow ES_P(Y) \colon Y \subseteq At(P)\}$
- X is a model of $cmpl^{\leftarrow}(P) \cup \{Y^{\vee} \rightarrow ES_P(Y) \colon Y \subseteq At(P)\}$
- OK to replace $cmpl^{\leftarrow}(P)$ with $cmpl(P)$
  - $cmpl^{\rightarrow}(P) \subseteq \{Y^{\wedge} \rightarrow ES_P(Y) \colon Y \subseteq At(P)\}$
  - $cmpl^{\rightarrow}(P) \subseteq \{Y^{\vee} \rightarrow ES_P(Y) \colon Y \subseteq At(P)\}$

Loops

Definition

- A loop is a set $Y \subseteq At(P)$ that induces in $G^+(P)$ a strongly connected subgraph
- In particular, all singleton sets are loops

Loops — example

Program P

$$
\begin{array}{lll}
p & \leftarrow & q, not\ r \\
q & \leftarrow & p \\
r & \leftarrow & not\ p
\end{array}
$$

Graph $G^+(P)$

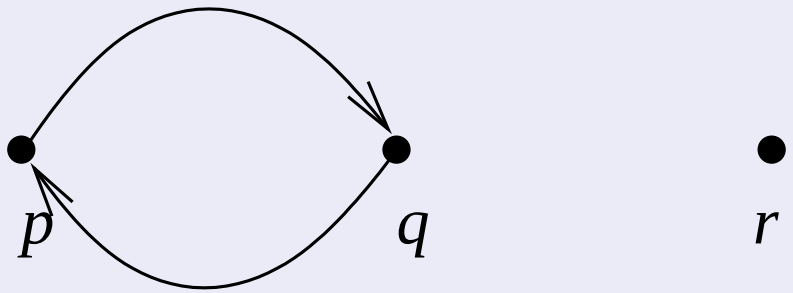
- $\{p\}, \{q\}, \{r\}, \{p, q\}$ are loops
- $\{p, q, r\}$ is not!

Loops — example

Program P

$$
\begin{array}{lll}
p & \leftarrow & q, not\ r \\
q & \leftarrow & p \\
r & \leftarrow & not\ p
\end{array}
$$

Graph $G^+(P)$



- $\{p\}, \{q\}, \{r\}, \{p, q\}$ are loops
- $\{p, q, r\}$ is not!

Loop Theorem

The following conditions are equivalent

- X is a stable model of P
- X is a model of $cmpl^{\leftarrow}(P) \cup \{Y^{\wedge} \rightarrow ES_P(Y) \colon Y$ – a loop$\}$
- X is a model of $cmpl^{\leftarrow}(P) \cup \{Y^{\vee} \rightarrow ES_P(Y) \colon Y$ – a loop$\}$
- OK to replace $cmpl^{\leftarrow}(P)$ with $cmpl(P)$
  - Singleton sets are loops!

Program inconsistency

Some programs have no stable nor supported models

- Sufficient conditions for the existence of stable models
- 4-val supported and stable models

Sufficient conditions

General dependency graph $G(P)$

- Atoms of P are vertices
- (a, b) is an edge in P if for some $r \in P$: $hd(r) = a, b \in bd(r)$
- If $b \in bd^+(r)$ — edge is positive
- If $b \in bd^-(r)$ — edge is negative

A propositional program P is

- Call-consistent: if $G(P)$ has no odd cycles (cycles with an odd number of negative edges)
- Stratified: if $G(P)$ has no paths with infinitely many negative edges
  - in particular, no cycles with a negative edge (for finite programs both conditions are equivalent)

Sufficient conditions

General dependency graph $G(P)$

- Atoms of P are vertices
- (a, b) is an edge in P if for some $r \in P$: $hd(r) = a, b \in bd(r)$
- If $b \in bd^+(r)$ — edge is positive
- If $b \in bd^-(r)$ — edge is negative

A propositional program P is

- Call-consistent: if $G(P)$ has no odd cycles (cycles with an odd number of negative edges)
- Stratified: if $G(P)$ has no paths with infinitely many negative edges
  - in particular, no cycles with a negative edge (for finite programs both conditions are equivalent)

Sufficient conditions

Results

- If a finite logic program is call-consistent, it has a stable model
- If a program is stratified it has a unique stable model

Equivalence — logics behind nonmonotonic logics

What do I mean?

- Logic allows us to manipulate theories
- Tautologies can be added or removed without changing the meaning
- Consequences of formulas in theories can be added or removed without changing the meaning
  - $\{p, p \supset q\}$ the same as $\{p, p \supset q, q\}$
  - one can always be replaced with another (within any larger context)
- Equivalence for replacement — logical equivalence necessary and sufficient
- Is there a logic which captures such manipulation with theories in nonmonotonic systems?

Is it important?

Query optimization

- Compute answers to a query Q (program) from a knowledge base KB (another program)
  reason from $Q \cup KB$
- Rewrite Q into an equivalent query Q', which can be processed more efficiently
  reasoning from $Q' \cup KB$ *easier*
- When are two queries equivalent?
  - If $Q \cup KB$ and $Q' \cup KB$ have the same meaning
    not quite what we want — knowledge-base dependent
  - If $Q \cup KB$ and $Q' \cup KB$ have the same meaning for every knowledge base KB
    better — knowledge-base independent

Towards modular logic programming

Equivalence of programs

- P and Q are equivalent if they have the same models

Nonmonotonic equivalence of programs

- P and Q are stable-equivalent if they have the same stable models

Towards modular logic programming

Equivalence of programs

- P and Q are equivalent if they have the same models

Nonmonotonic equivalence of programs

- P and Q are stable-equivalent if they have the same stable models

Towards modular logic programming

Equivalence for replacement

- Equivalence for replacement — for every program R, programs $P \cup R$ and $Q \cup R$ have the same stable models
- Commonly known as strong equivalence

  Lifschitz, Pearce, Valverde 2001; Lin 2002; Turner 2003; Eiter, Fink 2003; Eiter, Fink, Tompits, Woltran, 2005; T_ 2006; Woltran 2008
- Different than equivalence
  - $\{p \leftarrow not\ q\}$ and $\{q \leftarrow not\ p\}$
  - The same models but different meaning
- Different than stable-equivalence
  - $P = \{p\}$ and $Q = \{p \leftarrow not\ q\}$
  - The same stable models; $\{p\}$ is the only stable model in each case
  - But, $P \cup \{q\}$ and $Q \cup \{q\}$ have different stable models! ($\{p, q\}$ and $\{q\}$, respectively)

When are two programs strongly equivalent?

Se-model characterization

- A pair (X, Y) of sets of atoms is an *se-model* of a program P if
  - $X \subseteq Y$
  - $Y \models P$
  - $X \models P^Y$
- $SE(P)$ set of se-models of P
- Logic programs P and Q are strongly equivalent iff they have the same se-models ($SE(P) = SE(Q)$)
  - A similar concept characterizes strong equivalence of default theories

    Turner 2003

An interesting variant

Uniform equivalence

- Programs P and Q are uniformly equivalent if for every set D of facts (rules with empty body) $P \cup D$ and $Q \cup D$ have the same stable models
- Relevant for DB query optimization
- Different than other types of equivalence discussed here

When are two programs uniformly equivalent?

Se-model characterization

- Programs P and Q are uniformly equivalent iff
  - for every $Y \subseteq At$, Y is a model of P if and only if Y is a model of Q
  - for every $(x, y) \in SE(P)$ such that $X \subset Y$, there is $U \subseteq At$ such that $X \subseteq U \subset Y$ and $(U, Y) \in SE(Q)$
  - for every $(x, y) \in SE(Q)$ such that $X \subset Y$, there is $U \subseteq At$ such that $X \subseteq U \subset Y$ and $(U, Y) \in SE(P)$

When are two programs uniformly equivalent?

Ue-model characterization

- A pair (X, Y) of sets of atoms is a *ue-model* of a program P if it is an se-model of P and
- For every se-model (X', Y) such that $X \subseteq X'$, $X' = X$ or $X' = Y$
- Finite logic programs P and Q are uniformly equivalent iff they have the same ue-models

  Eiter and Fink, 2003

Formulas

- Base: atoms and the symbol $\bot$ ("false")
- Connectives $\wedge$, $\vee$ and $\rightarrow$
- Shortcuts
  - $\neg F ::= F \rightarrow \bot$
  - $\top ::= \bot \rightarrow \bot$
  - $F \leftrightarrow G ::= (F \rightarrow G) \wedge (G \rightarrow F)$

General logic programs

Positive and negative occurrences of atoms in formulas

- An occurrence of a in F is positive, if the # of implications with this occurrence of a in antecedent is even
- Otherwise, it is negative
- An occurrence of a in F is strictly positive if no implication contains this occurrence of a in the antecedent
  - $\neg F$ (that is, $F \to \bot$) has no strict occurrences of any atom.
- A head atom (of a formula) an atom with at least one strictly positive occurrence
- In $(\neg p \to q) \to (p \vee \neg q)$:
  - the first occurrence of p is negative
  - the second occurrence of p is strictly positive
  - both occurrences of q are negative

Stable-model semantics

Reduct of a formula F with respect to a set X of atoms

- The formula F^X obtained by replacing in F each maximal subformula of F that is not satisfied by X with $\bot$

Example: $F = (\neg p \rightarrow q) \wedge (\neg q \rightarrow p)$ and $X = \{p\}$

- $\neg p = p \rightarrow \bot$, and $X \models \neg p \rightarrow q$
- Thus: $\neg p$ is a maximal subformula not satisfied by X
- $\neg q = q \rightarrow \bot$, $X \not\models q$, $X \models \neg q$
- Thus, q is a maximal subformula not satisfied by X
- Thus: $F^X = (\bot \rightarrow q) \wedge ((\bot \rightarrow \bot) \rightarrow p)$
- Classically equivalent to p

Stable-model semantics

Reduct of a formula F with respect to a set X of atoms

- The formula F^X obtained by replacing in F each maximal subformula of F that is not satisfied by X with $\bot$

Example: $F = (\neg p \rightarrow q) \wedge (\neg q \rightarrow p)$ and $X = \{p\}$

- $\neg p = p \rightarrow \bot$, and $X \models \neg p \rightarrow q$
- Thus: $\neg p$ is a maximal subformula not satisfied by X
- $\neg q = q \rightarrow \bot$, $X \not\models q$, $X \models \neg q$
- Thus, q is a maximal subformula not satisfied by X
- Thus: $F^X = (\bot \rightarrow q) \wedge ((\bot \rightarrow \bot) \rightarrow p)$
- Classically equivalent to p

Stable-model semantics

To facilitate computation of the reduct

- $\bot^X = \bot$
- For a an atom, if $a \in X$, $a^X = a$; otherwise, $a^X = \bot$
- If $X \models F \circ G$, $(F \circ G)^X = F^X \circ G^X$; otherwise, $(F \circ G)^X = \bot$ ($\circ$ stands for any of $\wedge$, $\vee$, $\rightarrow$)
- If $X \models F$, $(\neg F)^X = \bot$; otherwise, $(\neg F)^X = (F \rightarrow \bot) = (\bot \rightarrow \bot) = \top$

Stable-model semantics

Definition

- A set X of atoms is a *stable model* of a formula F if X is a minimal model of F

Example: $F = (\neg p \to q) \wedge (\neg q \to p), X = \{p\}$

- $F^X = (\bot \to q) \wedge ((\bot \to \bot) \to p)$ (which is equivalent to p)
- X is a minimal model of F^X, so a stable model

Example: $F = (\neg p \to q) \wedge (\neg q \to p), X = \{p, q\}$

- $F^X = (\bot \to q) \wedge (\bot \to p)$ (which is equivalent to $\top$)
- X is not a minimal model of F^X, so not a stable model

Stable-model semantics

Definition

- A set X of atoms is a *stable model* of a formula F if X is a minimal model of F

Example: $F = (\neg p \to q) \wedge (\neg q \to p), X = \{p\}$

- $F^X = (\bot \to q) \wedge ((\bot \to \bot) \to p)$ (which is equivalent to p)
- X is a minimal model of F^X, so a stable model

Example: $F = (\neg p \to q) \wedge (\neg q \to p), X = \{p, q\}$

- $F^X = (\bot \to q) \wedge (\bot \to p)$ (which is equivalent to $\top$)
- X is not a minimal model of F^X, so not a stable model

Stable-model semantics

Definition

- A set X of atoms is a *stable model* of a formula F if X is a minimal model of F

Example: $F = (\neg p \to q) \wedge (\neg q \to p), X = \{p\}$

- $F^X = (\bot \to q) \wedge ((\bot \to \bot) \to p)$ (which is equivalent to p)
- X is a minimal model of F^X, so a stable model

Example: $F = (\neg p \to q) \wedge (\neg q \to p), X = \{p, q\}$

- $F^X = (\bot \to q) \wedge (\bot \to p)$ (which is equivalent to $\top$)
- X is not a minimal model of F^X, so not a stable model

Stable-model semantics

Properties

- If X is a stable model of a formula F then X consists of head atoms of F
- A least model of a Horn formula (conjunction of definite Horn clauses given as implications) is a unique stable model of the theory
- A set X is a stable model of a formula $F \wedge \neg G$ if and only if X is a stable model of F and $X \models \neg G$

Stable-model semantics

Strong equivalence

- Formulas F and F' are strongly equivalent if for every formula G, $F \wedge G$ and $F' \wedge G$ have the same stable models
- (X, Y) is an *se-model* of F if $Y \subseteq At$, $X \subseteq Y$, $Y \models F$ and $X \models F^Y$.
- The following conditions are equivalent:
  - Formulas F and G are strongly equivalent
  - For every set X of atoms, F^X and G^X are equivalent in classical logic
  - F and G have the same se-models
  - F and G are equivalent in the logic here-and-there (details later)

Stable-model semantics

Splitting

- Let F and G be formulas such that F does not contain any of the head atoms of G
- A set X is a stable model of $F \wedge G$ iff there is a stable model Y of F such that X is a stable model of $G \wedge \bigwedge Y$

Multivalued semantics

2-input one-step operator Φ_P

- Given two interpretations I and J

$$\Phi_P(I,J) = \{hd(r) : r \in P,\ bd^+(r) \subseteq I,\ bd^-(r) \cap J = \emptyset\}$$

- $\Phi_P(\cdot, J)$ monotone
- $\Phi_P(I, \cdot)$ antimonotone
- $\Phi_P(I, I) = T_P(I)$

Multivalued semantics

4-val interpretations

- Pairs (I, J) of interpretations
- Atoms in I are known and atoms in J are possible
- Give rise to 4 truth values
  - If $a \in I \cap J$, a is true
  - If $a \notin I \cup J$, a is false
  - If $a \in J \setminus I$, a is unknown
  - If $a \in I \setminus J$, a is overdefined (inconsistent)
- (I, J) consistent if $I \subseteq J$

Multivalued semantics

4-val one-step provability operator

- $\mathcal{T}_P(I,J) = (\Phi_P(I,J), \Phi_P(J,I))$
- Precision (information) ordering:
  $(I,J) \leq_i (I',J')$ - if $I \subseteq I'$ and $J' \subseteq J$
- $\mathcal{T}_P$ monotone wrt $\leq_i$
- $(I,J) \leq_i (I'J')$ $\Rightarrow$ $\mathcal{T}_P(I,J) \leq_i \mathcal{T}_P(I',J')$

4-val supported models

- (I, J) is a 4-val supported model of P if $(I, J) = \mathcal{T}_P(I, J)$
- (I, I) is a 4-val supported model iff I is a supported model
- The least 4-val supported model exists!
  - $\mathcal{T}_P$ is monotone and so has the least (wrt $\leq_i$) fixpoint
  - Moreover, it is consistent!
- Kripke-Kleene (Fitting) fixpoint or semantics: $(KK^t(P), KK^p(P))$

Well-founded semantics

van Gelder, Ross, Schlipf, 1991

- 4-val Gelfond-Lifschitz operator
- $\mathcal{GL}_P(I,J) = (GL_P(J), GL(I))$
- Also monotone wrt $\leq_i$
- (I,J) is a 4-val stable model if $\mathcal{GL}_P(I,J) = (I,J)$
- M is a stable model of P if and only if (M,M) is a 4-val stable model of P
- The least fixpoint of $\mathcal{GL}$ exists!! (by monotonicity)
- And is consistent
- Well-founded fixpoint (semantics): $(WF^t(P), WF^p(P))$
- For every stable model M of P

$$WF^t(P) \subseteq M \subseteq WF^p(P)$$

Logic here-and-there

Logic here-and-there, Heyting 1930

Syntax

- Connectives: $\bot$, $\vee$, $\wedge$, $\rightarrow$
- Formulas - standard extension of atoms by means of connectives
- $\neg\varphi$ - shorthand for $\varphi \rightarrow \bot$
- $\varphi \leftrightarrow \psi$ - shorthand for $(\varphi \rightarrow \psi) \wedge (\psi \rightarrow \varphi)$
- Language $\mathcal{L}_{ht}$

Logic here-and-there

Why important?

- Disjunctive logic programs — special theories in $\mathcal{L}_{ht}$
  - $a_1 | \ldots | a_k \leftarrow b_1, \ldots, b_m, not\ c_1, \ldots not\ c_n$
  - $b_1 \wedge \ldots \wedge b_m \wedge \neg c_1 \wedge \ldots \wedge \neg c_n \rightarrow c_1 \vee \ldots \vee c_n$
- General logic programs (Ferraris, Lifschitz) $=$ theories in $\mathcal{L}_{ht}$
  - answer-set semantics extends to general logic programs
  - equilibrium models in logic *ht*
  - the two coincide!

Entailment in logic here-and-there

Ht-interpretations

- Pairs $\langle H, T \rangle$, where $H \subseteq T$ are sets of atoms
- Kripke interpretations with two worlds "here" and "there"
  - H determines the valuation for "here"
  - T determines the valuation for "there"

Kripke-model satisfiability in the world "here" $\models_{ht}$

- $\langle H, T \rangle \not\models_{ht} \bot$
- $\langle H, T \rangle \models_{ht} p$ if $p \in H$ (for atoms only)
- $\langle H, T \rangle \models_{ht} \varphi \wedge \psi$ and $\langle H, T \rangle \models_{ht} \varphi \vee \psi$ — standard recursion
- $\langle H, T \rangle \models_{ht} \varphi \rightarrow \psi$ if
  - $\langle H, T \rangle \not\models_{ht} \varphi$ or $\langle H, T \rangle \models_{ht} \psi$
  - $T \models \varphi \rightarrow \psi$ (in standard propositional logic).

Entailment in logic here-and-there

Ht-interpretations

- Pairs $\langle H, T \rangle$, where $H \subseteq T$ are sets of atoms
- Kripke interpretations with two worlds "here" and "there"
  - H determines the valuation for "here"
  - T determines the valuation for "there"

Kripke-model satisfiability in the world "here" $\models_{ht}$

- $\langle H, T \rangle \not\models_{ht} \bot$
- $\langle H, T \rangle \models_{ht} p$ if $p \in H$ (for atoms only)
- $\langle H, T \rangle \models_{ht} \varphi \wedge \psi$ and $\langle H, T \rangle \models_{ht} \varphi \vee \psi$ — standard recursion
- $\langle H, T \rangle \models_{ht} \varphi \rightarrow \psi$ if
  - $\langle H, T \rangle \not\models_{ht} \varphi$ or $\langle H, T \rangle \models_{ht} \psi$
  - $T \models \varphi \rightarrow \psi$ (in standard propositional logic).

Entailment in logic here-and-there

ht-model, *ht*-validity, *ht*-equivalence

- If $\langle H, T \rangle \models_{ht} \varphi$ - $\langle H, T \rangle$ is an *ht-model* of φ
- φ is *ht-valid* if for every *ht*-model $\langle H, T \rangle$, $\langle H, T \rangle \models \varphi$
- φ and ψ are *ht-equivalent* if they have the same $ht-$models

- φ and ψ are ht-equivalent iff $\varphi \leftrightarrow \psi$ is *ht*-valid

Natural deduction — sequents and rules

- Sequents $\Gamma \Rightarrow \varphi$ — "φ under the assumptions Γ"
- Introduction rules for $\wedge$, $\vee$, $\rightarrow$

$$\frac{\Gamma \Rightarrow \varphi \quad \Delta \Rightarrow \psi}{\Gamma, \Delta \Rightarrow \varphi \wedge \psi}$$

- Elimination rules for $\wedge$, $\vee$, $\rightarrow$

$$\frac{\Gamma \Rightarrow \varphi \quad \Delta \Rightarrow \varphi \rightarrow \psi}{\Gamma, \Delta \Rightarrow \psi}$$

- Contradiction

$$\frac{\Gamma \Rightarrow \bot}{\Gamma \Rightarrow \varphi}$$

- Weakening

$$\frac{\Gamma \Rightarrow \varphi}{\Gamma' \Rightarrow \varphi} \qquad \text{for all } \Gamma', \Gamma \text{ s.t. } \Gamma' \subseteq \Gamma$$

Proof theory

Axiom schemas

(AS1)	$\varphi \Rightarrow \varphi$	
(AS2)	$\Rightarrow \varphi \vee \neg\varphi$	(Excluded Middle)
(AS2′)	$\Rightarrow \neg\varphi \vee \neg\neg\varphi$	(Weak EM)
(AS2″)	$\Rightarrow \varphi \vee (\varphi \rightarrow \psi) \vee \neg\psi$	(in between (AS2) and (AS2′)

Logics through natural deduction

Propositional logic	(AS1), (AS2)
Intuitionistic logic	(AS1)
Logic here-and-there	(AS1),(AS2″)

Proof theory

Axiom schemas

(AS1)	$\varphi \Rightarrow \varphi$	
(AS2)	$\Rightarrow \varphi \vee \neg\varphi$	(Excluded Middle)
(AS2′)	$\Rightarrow \neg\varphi \vee \neg\neg\varphi$	(Weak EM)
(AS2″)	$\Rightarrow \varphi \vee (\varphi \rightarrow \psi) \vee \neg\psi$	(in between (AS2) and (AS2′)

Logics through natural deduction

Propositional logic	(AS1), (AS2)
Intuitionistic logic	(AS1)
Logic here-and-there	(AS1),(AS2″)

Bringing the two together

Soundness and completeness

- A formula is a theorem of *ht* if and only if it is *ht*-valid

In particular

- φ and ψ are *ht*-equivalent iff $\Rightarrow \varphi \leftrightarrow \psi$ is a theorem of *ht*

Bringing the two together

Soundness and completeness

- A formula is a theorem of *ht* if and only if it is *ht*-valid

In particular

- φ and ψ are *ht*-equivalent iff $\Rightarrow \varphi \leftrightarrow \psi$ is a theorem of *ht*

Logic here-and-there and ASP

Equilibrium models, Pearce 1997

- $\langle T, T \rangle$ is an *equilibrium model* of a set A of formulas if
  - $\langle T, T \rangle \models_{ht} A$, and
  - for every $H \subseteq T$ such that $\langle H, T \rangle \models_{ht} A$, $H = T$

Key connection

- A set M of atoms is an answer set of a disjunctive logic program P (general logic program P) if and only if $\langle M, M \rangle$ is an equlibrium model for P

Key application

Strong equivalence

- Let P and Q be two (general) programs. The following conditions are equivalent:
  - P and Q are strongly equivalent
  - P and Q are *ht*-equivalent
  - P and Q have the same *ht*-models
  - $P \leftrightarrow Q$ is *ht*-valid
  - $\Rightarrow P \leftrightarrow Q$ is a theorem of *ht*

Modal logics

Modal logics

The language $\mathcal{L}_L$

- $\varphi ::= \bot \mid p \mid L\varphi \mid \neg\varphi \mid \varphi \vee \varphi \mid \varphi \wedge \varphi \mid \varphi \supset \varphi$ (where p - an atom)
- $a \supset L(\neg b \wedge L(a \vee \neg b))$
- A language to express "modalities"
  - "is known"
  - "is believed"
  - "is possible"
  - "is provable"
  - "is necessary"
  - ...
- We will read $L\varphi$ as: φ is defeasibly known
- Common abbreviation: $M\varphi ::= \neg L\neg\varphi$
  - $M\varphi$ - "φ is possible"

Modal logics

Proof theory

- Modus ponens and necessitation $\frac{\varphi}{L\varphi}$
- Instances of propositional tautologies: $L\varphi \vee \neg L\varphi$, etc.
- Instances of modal axiom schemata such as:
  - K: $L(\varphi \supset \psi) \supset (L\varphi \supset L\psi)$
  - T: $L\varphi \supset \varphi$
  - 4: $L\varphi \supset LL\varphi$
  - F: $(\varphi \wedge \neg L\neg L\psi) \supset L(\neg\varphi \vee \psi)$
  - W5: $\varphi \wedge ML\varphi \supset L\varphi$
  - 5: $\neg L\neg L\varphi \supset L\varphi$ (or $\neg L\varphi \supset L\neg L\varphi$)
- Logics determined by modal axioms
  - Modal logic S4: K, T, 4
  - Modal logic S4F: K, T, 4, F
  - Modal logic SW5: K, T, 4,
  - Modal logic S5: K, T, 4, 5

Modal logics: proof theory example

Lemma: if $\vdash \varphi \supset \psi$ then $\vdash L\varphi \supset L\psi$

1.	$L(\varphi \rightarrow \psi) \supset (L\varphi \supset L\psi)$	axiom **K**
2.	$\varphi \supset \psi$	given
3.	$L(\varphi \supset \psi)$	necessitation
4.	$L\varphi \supset L\psi$	modus ponens

Modal logics: proof theory example

Proof theory: $M\varphi \leftrightarrow MM\varphi$ is a theorem of S4

1.	$L\neg\neg L\neg\varphi \supset \neg\neg L\neg\varphi$	axiom **T**
2.	$L\neg\neg L\neg\varphi \supset L\neg\varphi$	prop taut
3.	$\neg L\neg\varphi \supset \neg L\neg\neg L\neg\varphi$	prop taut
4.	$M\varphi \supset MM\varphi$	rewriting
5.	$L\neg\varphi \supset LL\neg\varphi$	axiom **4**
6.	$L\neg\varphi \supset \neg\neg L\neg\varphi$	prop taut
7.	$LL\neg\varphi \supset L\neg\neg L\neg\varphi$	lemma
8.	$L\neg\varphi \supset L\neg\neg L\neg\varphi$	from (5) and (7)
9.	$\neg L\neg\neg L\neg\varphi \supset \neg L\neg\varphi$	prop taut
10.	$MM\varphi \supset M\varphi$	rewriting

Modal logics

More theorems

1.	$MM\varphi \leftrightarrow M\varphi$	thm of S4
2.	$LL\varphi \leftrightarrow L\varphi$	thm of S4
3.	$LMLM\varphi \leftrightarrow LM\varphi$	thm of S4
4.	$MLML\varphi \leftrightarrow ML\varphi$	thm of S4
5.	$LML\varphi \leftrightarrow ML\varphi$	thm of S4F
6.	$MLM\varphi \leftrightarrow LM\varphi$	thm of S4F

Modal logics

Modalities

- Sequences of modal operators: L, MML, $MMMLMMML$, etc
- L, M, LM, ML, LML, MLM — the only nontrivial modalities of S4
- L, M, LM, ML — the only nontrivial modalities of S4F
- But: $M = \neg L \neg$ and $\neg LM = ML \neg$
- So, only two different modal notions modeled in S4F:
  - L - “defeasibly knowing”
  - ML - “believing”

Modal logics

Soundness and completeness for the logic K

- φ is a theorem of the logic K if and only if φ is valid in every Kripke model
- Logic K is characterized by the class of all Kripke models

Soundness and completeness for other logics

- Restrictions on the form of the accessibility relation needed
- Logic $\mathcal{S}$ is characterized by the class of Kripke models with with the accessibility relation satisfying properties . . .

Modal logics

Soundness and completeness for the logic K

- φ is a theorem of the logic K if and only if φ is valid in every Kripke model
- Logic K is characterized by the class of all Kripke models

Soundness and completeness for other logics

- Restrictions on the form of the accessibility relation needed
- Logic $\mathcal{S}$ is characterized by the class of Kripke models with with the accessibility relation satisfying properties . . .

Modal nonmonotonic logics

Expansions

- $\mathcal{S}$ — modal (monotone) logic
- $\mathcal{S}$-expansion of a modal theory $T \subseteq \mathcal{L}_L$:

$$E = Cn_{\mathcal{S}}(\{T \cup \{\neg L\varphi | \varphi \in \mathcal{L}_L \setminus E\})$$

Relation to answer sets?

- M is an answer set of a (disjunctive) LP P if and only if
- $M = LM(P \cup \{not\ a \mid a \in At \setminus M\}) \cap At$
- $M \in LM(P \cup \{not\ a \mid a \in At \setminus M\}) \cap At$ - for disjunctive LPs
  - $not\ a$ is treated as a new propositional atom

Modal nonmonotonic logics

Expansions

- $\mathcal{S}$ — modal (monotone) logic
- $\mathcal{S}$-expansion of a modal theory $T \subseteq \mathcal{L}_L$:

$$E = Cn_{\mathcal{S}}(\{T \cup \{\neg L\varphi | \varphi \in \mathcal{L}_L \setminus E\})$$

Relation to answer sets?

- M is an answer set of a (disjunctive) LP P if and only if
- $M = LM(P \cup \{not\ a \,|\, a \in At \setminus M\}) \cap At$
- $M \in LM(P \cup \{not\ a \,|\, a \in At \setminus M\}) \cap At$ - for disjunctive LPs
  - $not\ a$ is treated as a new propositional atom

Modal nonmonotonic logics

Properties of expansions: T - an expansion

- Expansions are closed under S5
  - $T = Cn_{S5}(T)$
- Expansions are epistemically complete
  - for every $\varphi \in \mathcal{L}_L$: either $L\varphi \in T$ or $\neg L\varphi \in T$
- Expansions are closed under introspection
  - if $\varphi \in T$ then $L\varphi \in T$
  - if $\varphi \notin T$ then $\neg L\varphi \in T$

Representation of expansions

- If T is an expansion, $T = [A]$, for some $A \subseteq \mathcal{L}$
- Here $[A]$ is a theory of a certain Kripke model determined by A

Modal nonmonotonic logics

Properties of expansions: T - an expansion

- Expansions are closed under S5
  - $T = Cn_{S5}(T)$
- Expansions are epistemically complete
  - for every $\varphi \in \mathcal{L}_L$: either $L\varphi \in T$ or $\neg L\varphi \in T$
- Expansions are closed under introspection
  - if $\varphi \in T$ then $L\varphi \in T$
  - if $\varphi \notin T$ then $\neg L\varphi \in T$

Representation of expansions

- If T is an expansion, $T = [A]$, for some $A \subseteq \mathcal{L}$
- Here $[A]$ is a theory of a certain Kripke model determined by A

Modal nonmonotonic logics

Nonmonotonic S4F captures (T_, 1991; Schwarz and T_, 1994)

- (Disjunctive) logic programming with the answer set semantics

$$La_1 \vee \ldots \vee La_k \leftarrow Lb_1, \ldots, Lb_m, \neg MLc_1, \ldots, \neg MLc_n$$

- (Disjunctive) default logic
- General default logic *(Cabalar, 2004; extended by T_, 2007)*
- Logic of grounded knowledge *(Lin and Shoham, 1990)*
- Logic of minimal belief and negation as failure *(Lifschitz, 1994)*
- Is S4F the logic underlying nonmon reasoning?

Theory simplifies some restrictions

Modal defaults and modal default theories

- $\varphi ::= L\psi \mid L\varphi \mid \neg\varphi \mid \varphi \vee \varphi \mid \varphi \wedge \varphi \mid \varphi \supset \varphi$
  where ψ — a propositional formula
- For modal default theories (sets of modal defaults) S4F characterizes strong equivalence!

First, the semantics simplifies!

Se-interpretations

- $\langle H, T \rangle$ - H, T are propositional theories closed under propositional entailment
- Entailment relations $\models_t$ and $\models_h$ for modal defaults
- $\langle H, T \rangle \models_t \varphi$
  - $\varphi = L\psi$ (ψ is propositional): $\langle H, T \rangle \models_t \varphi$ if $\psi \in T$
  - Boolean connectives standard
  - $\varphi = L\psi$, where ψ is a modal default
    $\langle H, T \rangle \models_t \varphi$ if $\langle H, T \rangle \models_t \psi$
- We write $\langle H, T \rangle \models \varphi$ if $\langle H, T \rangle \models_h \varphi$ and $\langle H, T \rangle \models_t \varphi$

First, the semantics simplifies!

Se-interpretations

- $\langle H, T \rangle$ - H, T are propositional theories closed under propositional entailment
- Entailment relations $\models_t$ and $\models_h$ for modal defaults
- $\langle H, T \rangle \models_h \varphi$
  - $\varphi = L\psi$ (where ψ is propositional): $\langle H, T \rangle \models_h \varphi$ if $\psi \in H$
  - Boolean connectives standard
  - $\varphi = L\psi$, where ψ is a modal default

    $\langle H, T \rangle \models_h \varphi$ if $\langle H, T \rangle \models_h \psi$ and $\langle H, T \rangle \models_t \psi$
- We write $\langle H, T \rangle \models \varphi$ if $\langle H, T \rangle \models_h \varphi$ and $\langle H, T \rangle \models_t \varphi$

First, the semantics simplifies!

Se-interpretations

- $\langle H, T \rangle$ - H, T are propositional theories closed under propositional entailment
- Entailment relations $\models_t$ and $\models_h$ for modal defaults
- $\langle H, T \rangle \models_h \varphi$
  - $\varphi = L\psi$ (where ψ is propositional): $\langle H, T \rangle \models_h \varphi$ if $\psi \in H$
  - Boolean connectives standard
  - $\varphi = L\psi$, where ψ is a modal default
    $\langle H, T \rangle \models_h \varphi$ if $\langle H, T \rangle \models_h \psi$ and $\langle H, T \rangle \models_t \psi$
- We write $\langle H, T \rangle \models \varphi$ if $\langle H, T \rangle \models_h \varphi$ and $\langle H, T \rangle \models_t \varphi$

Putting it all together

Se-interpretations $\equiv$ se-models (for modal defaults)

- Under the restriction to modal defaults and modal default theories, se-interpretations characterize the entailment relation in S4F
- E is an extension of I iff $\langle E, E\rangle \models P$ and for every $E' \subseteq E$ such that $\langle E', E\rangle \models P$, $E' = E$
- Generalizes the concept of an extension introduced by Reiter
- Similarities with equilibrium models

Putting it all together

Strong equivalence

- Let $I', I'' \subseteq \mathcal{L}_L$ be modal defalt theories. The following conditions are equivalent:
  - I' and I'' are strongly equivalent ($I' \cup I$ and $I'' \cup I$ have the same S4F-expansions for every modal default theory I)
  - I' and I'' are valid in the same se-interpretations

General logic programming

Modal rules, modal programs

- Modal rule: $\varphi ::= Lp \mid L\varphi \mid \neg\varphi \mid \varphi \vee \varphi \mid \varphi \wedge \varphi \mid \varphi \supset \varphi$ where p is a propositional atom
- A special class of modal default theories
- SW5 can be used instead of S4F
- Simple se-models: pairs $\langle H, T \rangle$, where H and T are sets of atoms, $H \subseteq T$
- M is an answer set of a modal program P if $\langle Cn(M), Cn(M) \rangle$ is a selected SW5-model of P
- M is an answer set of P iff $\langle M, M \rangle \models P$ and for every $M' \subseteq M$ such that $\langle M', M \rangle \models P$, $M' = M$
- Generalizes the concept of an answer set of a DLP by Gelfond and Lifschitz

General logic programming

Strong equivalence

- Let $P', P'' \subseteq \mathcal{L}_L$ be modal programs. The following conditions are equivalent:
  - P' and P'' are strongly equivalent
  - P' and P'' are valid in the same simple se-models
  -

Examples

- $H \leftarrow p, not\, p, B$ - a "tautology"
- $Lp \wedge \neg MLp \supset \bot$ - a thm of T (and so SW5)

General logic programming

Strong equivalence

- Let $P', P'' \subseteq \mathcal{L}_L$ be modal programs. The following conditions are equivalent:
  - P' and P'' are strongly equivalent
  - P' and P'' are valid in the same simple se-models
  -

Examples

- $H \leftarrow p, not\, p, B$ - a "tautology"
- $Lp \wedge \neg MLp \supset \bot$ - a thm of T (and so SW5)

Lessons

Logic here-and-there

- Is the logic of strong equivalence in general logic programming
- Characterizes uniform equivalence in general logic programming
- Non-mon here-and-there = general LP *(Ferraris and Lifschitz)*

Lessons, cont'd

SW5 when restricted to modal programs

- An alternative to logic here-and-there
- Connectives "classical" (but modality in the language)
- Every theorem in SW5 yields a "tautology" for modal programs
- Supports transformations preserving answer sets
- But: is it the case that P and P' are strongly equivalent iff $P \leftrightarrow P'$ is a theorem of SW5?
  - For some classes of modal programs YES

Lessons, cont'd

S4F when restricted to modal defaults

- Extends SW5 (modal defaults properly extend modal programs)
- Captures several additional nonmonotonic logics
- Is the logic of strong equivalence in these formalisms
- As before, connectives "classical" (but modality in the language)
- Source of "tautologies"
- Supports transformations preserving extensions
- But: is it the case that φ and ψ are strongly equivalent iff $\varphi \leftrightarrow \psi$ is a theorem S4F?

Algebraic approach

The problem

Complex landscape of nonmonotonicity

- Multitude of formalisms
- Different intuitions
- Different languages
- Different semantics
- Complexity

Needed!

- Unifying abstract foundation

The problem

Complex landscape of nonmonotonicity

- Multitude of formalisms
- Different intuitions
- Different languages
- Different semantics
- Complexity

Needed!

- Unifying abstract foundation

A triumph of universal algebra

Basic lesson for this segment

- Major nonmonotonic systems
  - logic programming
  - default logic
  - autoepistemic logics

  can be given a unified algebraic treatment
- Each system can be assigned the same family of semantics
- Key concepts: lattices and bilattices, operators and fixpoints
- Key ideas: approximating operators and stable operators
- Key tool: Knaster-Tarski Theorem

Overview of approach

Generalize Fitting's work on logic programming

- Central role of 4-valued van Emden-Kowalski operator $\mathcal{T}_P$
- Derived stable operator, Ψ'_P
- 2-valued and 3-valued supported models and Kripke-Kleene semantics described by fixpoints of $\mathcal{T}_P$
- 2-valued and 3-valued stable models and well-founded semantics described by fixpoints of Ψ'_P

Lattices

Key definitions, some notation

- $\langle L, \leq \rangle$
  - L is a nonempty set
  - $\leq$ is a partial order such that every two lattice elements have *lub* (join) and *glb* (meet)
- Elements of L express
  - degree of truth
  - measure of knowledge
- $\leq$ - order of increased truth or knowledge
- Complete lattices (both bounds defined for all sets)
- $\bot$, $\top$

Lattices - examples

Lattice $\mathcal{TWO}$

- $\{\mathbf{f}, \mathbf{t}\}$
- $\mathbf{f} \leq \mathbf{t}$

Lattice $\mathcal{A}_2$

- set of all 2-valued interpretations
- componentwise extension of the ordering from $\mathcal{TWO}$

Lattice $\mathcal{W}$

- family of sets of 2-valued interpretations
- $W_1 \sqsubseteq W_2$ if $W_2 \subseteq W_1$

Lattices - examples

Lattice $\mathcal{TWO}$

- $\{\mathbf{f}, \mathbf{t}\}$
- $\mathbf{f} \leq \mathbf{t}$

Lattice $\mathcal{A}_2$

- set of all 2-valued interpretations
- componentwise extension of the ordering from $\mathcal{TWO}$

Lattice $\mathcal{W}$

- family of sets of 2-valued interpretations
- $W_1 \sqsubseteq W_2$ if $W_2 \subseteq W_1$

Lattices - examples

Lattice $\mathcal{TWO}$

- $\{\mathbf{f}, \mathbf{t}\}$
- $\mathbf{f} \leq \mathbf{t}$

Lattice $\mathcal{A}_2$

- set of all 2-valued interpretations
- componentwise extension of the ordering from $\mathcal{TWO}$

Lattice $\mathcal{W}$

- family of sets of 2-valued interpretations
- $W_1 \sqsubseteq W_2$ if $W_2 \subseteq W_1$

Operators

That's what it's all about!

- Truth or knowledge can be revised
- Revisions are described by operators on lattices
- Fixpoints — states of truth or knowledge that cannot be revised

Operators

Monotone operators

- An operator O is monotone if $x \leq y$ implies $O(x) \leq O(y)$
- Knaster-Tarski Theorem: a monotone operator on a complete lattice has a least fixpoint

Operators, cont'd

Antimonotone operators

- An operator O is antimonotone if $x \leq y$ implies $O(y) \leq O(x)$
- If O is antimonotone then O^2 is monotone:

$$x \leq y \;\Rightarrow\; O(y) \leq O(x) \;\Rightarrow\; O^2(x) \leq O^2(y)$$

- Oscillating pair: (x, y) is an oscillating pair for an operator O if $O(x) = y$ and $O^2(x) = x$
- Antimonotone operator O has an *extreme* oscillating pair

$$(lfp(O^2), gfp(O^2))$$

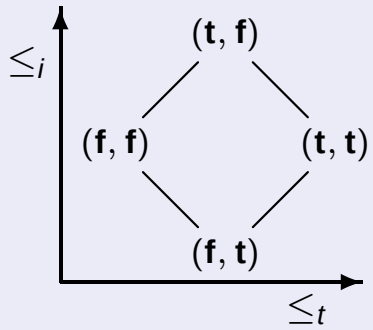
Approximations and bilattices

Key definitions, some notation

- A pair (x, y) approximates an element z if $x \leq z \leq y$
- Orderings of approximations:
  - information (or precision) ordering: $(x_1, y_1) \leq_i (x_2, y_2)$ iff $x_1 \leq x_2$ and $y_2 \leq y_1$
  - truth ordering: $(x_1, y_1) \leq_t (x_2, y_2)$ iff $x_1 \leq x_2$ and $y_1 \leq y_2$
- Bilattice $\langle L^2, \leq_i, \leq_t \rangle$
- A pair (x, y) is consistent if $x \leq y$, and inconsistent, otherwise
- An element (x, y) is complete if $x = y$

Bilattices - examples

Bilattice $\mathcal{FOUR}$

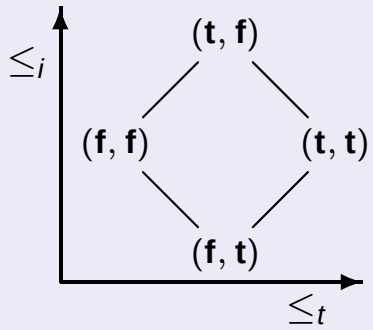


Bilattice $\mathcal{A}_4$

- set of all pairs of 2-valued interpretations (identified with 4-valued interpretations)
- componentwise extension of the orderings from $\mathcal{FOUR}$

Bilattices - examples

Bilattice $\mathcal{FOUR}$



Bilattice $\mathcal{A}_4$

- set of all pairs of 2-valued interpretations (identified with 4-valued interpretations)
- componentwise extension of the orderings from $\mathcal{FOUR}$

Bilattices - examples, cont'd

Bilattice $\mathcal{B}$

- Family of pairs of sets of 2-valued interpretations
- *Belief pairs*
- $(P_1, S_1) \sqsubseteq_i (P_2, S_2)$ if $P_2 \subseteq P_1$ and $S_1 \subseteq S_2$
- $(P_1, S_1) \sqsubseteq_t (P_2, S_2)$ if $P_2 \subseteq P_1$ and $S_2 \subseteq S_1$

Approximating operators

Key definitions, some notation

- $A : L^2 \to L^2$ approximates $O : L \to L$ if
  - $A(x, x) = (O(x), O(x))$
  - A is $\leq_i$-monotone
  - A is symmetric: $A^1(x, y) = A^2(y, x)$, where $A(x, y) = (A^1(x, y), A^2(x, y))$

Properties

- Approximating operators are consistent
- Complete fixpoints of A correspond to fixpoints of O
- Every fixpoint of A is approximated by the least fixpoint of A: Kripke-Kleene fixpoint of A
- Kripke-Kleene fixpoint of an approximating operator is consistent

Approximating operators

Key definitions, some notation

- $A : L^2 \to L^2$ approximates $O : L \to L$ if
  - $A(x, x) = (O(x), O(x))$
  - A is $\leq_i$-monotone
  - A is symmetric: $A^1(x, y) = A^2(y, x)$, where $A(x, y) = (A^1(x, y), A^2(x, y))$

Properties

- Approximating operators are consistent
- Complete fixpoints of A correspond to fixpoints of O
- Every fixpoint of A is approximated by the least fixpoint of A: Kripke-Kleene fixpoint of A
- Kripke-Kleene fixpoint of an approximating operator is consistent

Getting down to business!

Stable operators

- If $A : L^2 \to L^2$ is $\leq_i$-monotone then $A^1(\cdot, y)$ and $A^2(x, \cdot)$ are monotone
- For $\leq_i$-monotone operator $A : L^2 \to L^2$ define:

$$C_A^l(y) = lfp(A^1(\cdot, y)) \quad \text{and} \quad C_A^u(x) = lfp(A^2(x, \cdot))$$

- Since A is symmetric, $C_A^l = C_A^u = C_A$
- Stable operator for A:

$$\mathcal{C}_A(x, y) = (C_A(y), C_A(x))$$

- Stable fixpoints (relative to $\mathcal{C}_A$)
- $\leq_i$-least fixpoint of $\mathcal{C}_A$ — well-founded (WF) fixpoint of A

Properties of stable operators

All quite easy to prove, in fact

- C_A is antimonotone
- $\mathcal{C}_A$ is $\leq_i$-monotone and $\leq_t$-antimonotone
- Fixpoints of $\mathcal{C}_A$ are $\leq_t$-minimal fixpoints of A
- Complete fixpoints of $\mathcal{C}_A$ correspond to fixpoints of C_A
- Complete fixpoints of $\mathcal{C}_A$ are fixpoints of O
- K-K fixpoint of $A \leq_i$ WF fixpoint of A

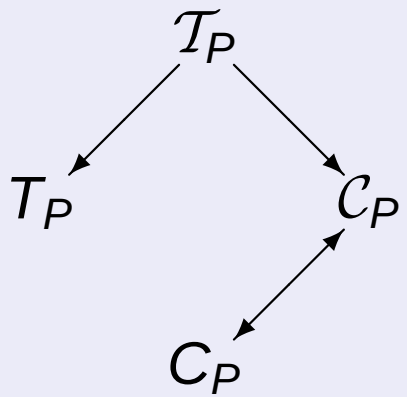
Logic programming — case study 1

Fitting

- Lattice $\mathcal{A}_2$, bilattice $\mathcal{A}_4$
- Operators associated with program P
  - 2-valued van Emden-Kowalski operator T_P
  - Its approximation: 4-valued van Emden-Kowalski operator $\mathcal{T}_P$
  - 2-valued stable operator (Gelfond-Lifschitz operator GL_P)
  - Stable operator $\mathcal{C}_P$ of $\mathcal{T}_P$ (operator Ψ'_P of Przymusinski)
- Semantics
  - Supported models: fixpoints of the operator $\mathcal{T}_P$ (T_P)
  - Kripke-Kleene semantics: least fixpoint of $\mathcal{T}_P$
  - Stable models: fixpoints of the operator $\mathcal{C}_P$ (C_P)
  - Well-founded semantics: least fixpoint of $\mathcal{C}_P$

Logic programming explained

Central role of $\mathcal{T}_P$



Autoepistemic Logic — case study 2

Truth assignment function $\mathcal{H}_{V,I}$

- For atom p: $\mathcal{H}_{V,I}(p) = I(p)$
- The boolean connectives — standard way
- $\mathcal{H}_{V,I}(KF) = \mathbf{t}$, if for every $J \in V$, $\mathcal{H}_{V,J}(F) = \mathbf{t}$
- $\mathcal{H}_{V,I}(KF) = \mathbf{f}$, otherwise

AE models, expansions

- Moore's operator $D_T : \mathcal{W} \to \mathcal{W}$

$$D_T(V) = \{I : \mathcal{H}_{V,I}(T) = \mathbf{t}\}$$

- Fixpoints of D_T — autoepistemic models of T
- Autoepistemic models generate *expansions*

Autoepistemic Logic — case study 2

Truth assignment function $\mathcal{H}_{V,I}$

- For atom p: $\mathcal{H}_{V,I}(p) = I(p)$
- The boolean connectives — standard way
- $\mathcal{H}_{V,I}(KF) = \mathbf{t}$, if for every $J \in V$, $\mathcal{H}_{V,J}(F) = \mathbf{t}$
- $\mathcal{H}_{V,I}(KF) = \mathbf{f}$, otherwise

AE models, expansions

- Moore's operator $D_T : \mathcal{W} \to \mathcal{W}$

$$D_T(V) = \{I : \mathcal{H}_{V,I}(T) = \mathbf{t}\}$$

- Fixpoints of D_T — autoepistemic models of T
- Autoepistemic models generate *expansions*

AEL — approximating operators

The setting

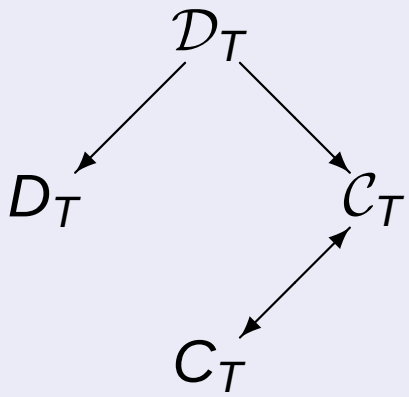
- Lattice $\mathcal{W}$, bilattice $\mathcal{B}$
- $\mathcal{H}^4_{(V,V'),I}$
- Approximating operator for D_T — $\mathcal{D}_T$ (DMT 98)

$$\mathcal{D}_T(V,V') = (\{I : \mathcal{H}^4_{(V,V'),I}(T) \geq_t (\mathbf{f},\mathbf{t})\}, \{I : \mathcal{H}^4_{(V,V'),I}(T) \geq_t (\mathbf{t},\mathbf{f})\})$$

- Complete fixpoints of $\mathcal{D}_T$ — autoepistemic models of T
- The least fixpoint of $\mathcal{D}_T$ — Kripke-Kleene fixpoint
  - approximates all autoepistemic models of T
- The stable operator for $\mathcal{D}_T$: $\mathcal{C}_T(V,V') = (C_T(V'), C_T(V))$
- What are the fixpoints of C_T?

Autoepistemic logic explained

Central role of $\mathcal{D}_T$



Default Logic — case study 3

Same setting as for AEL

- Lattice $\mathcal{W}$, bilattice $\mathcal{B}$
- $\mathcal{H}_{V,I}(\varphi) = I(\varphi)$, for every formula φ
- $d = \frac{\alpha : \beta_1, \ldots, \beta_k}{\gamma}$
- $\mathcal{H}_{V,I}(d) = \mathbf{t}$ iff
  - there is $J \in V$ such that $J(\alpha) = \mathbf{f}$, or
  - there is i, $1 \leq i \leq k$ such that for every $J \in V$, $J(\beta_i) = \mathbf{f}$, or
  - $I(\gamma) = \mathbf{t}$
- Weak-extension operator E_Δ (Δ — default theory):

$$E_\Delta(V) = \{I \in \mathcal{A}_2 : \mathcal{H}_{V,I}(\Delta) = \mathbf{t}\}$$

- Fixpoints of $E_\Delta(V)$ — default models of weak extensions of Δ

4-valued truth assignment, approximating operator

- $\mathcal{H}^4_{(V,V'),I}$
- Approximating operator for E_Δ — $\mathcal{E}_\Delta$

$$\mathcal{E}_\Delta(V, V') = (\{I : \mathcal{H}^4_{(V,V'),I}(\Delta) \geq_t (\mathbf{f}, \mathbf{t})\}, \{I : \mathcal{H}^4_{(V,V'),I}(\Delta) \geq_t (\mathbf{t}, \mathbf{f})\})$$

- Complete fixpoints of $\mathcal{E}_\Delta$ — models of weak extensions of Δ
- The least fixpoint of $\mathcal{E}_\Delta$ — Kripke-Kleene fixpoint
  - approximates all default models of weak extensions of Δ

Stable operator

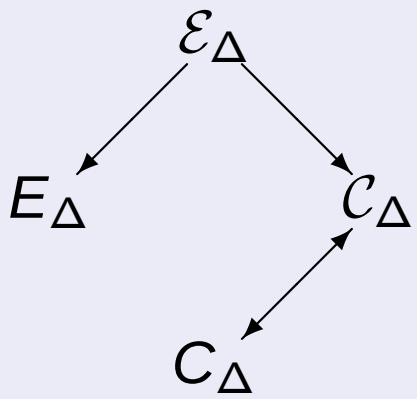
- The stable operator for $\mathcal{E}_\Delta$:

$$\mathcal{C}_\Delta(V, V') = (C_\Delta(V'), C_\Delta(V))$$

- C_Δ — Guerreiro-Casanova operator Σ_Δ
- Fixpoints of C_Δ — default models of Reiter's extensions
- Consistent fixpoints of $\mathcal{C}_\Delta$ — stationary extensions by Przymusinski
- Well-founded fixpoint of $\mathcal{E}_\Delta$ (least fixpoint of $\mathcal{C}_\Delta$ — well-founded semantics of default logic by Baral and Subrahmanian)

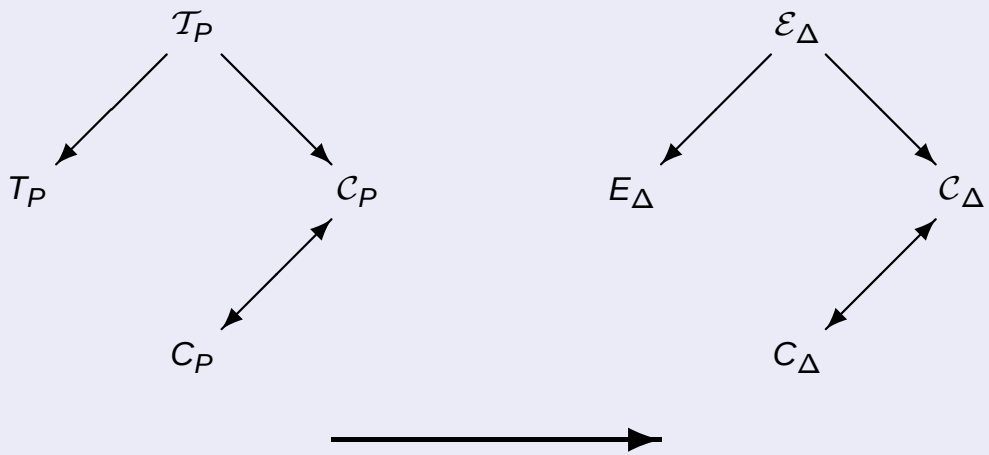
DL explained

Central role of $\mathcal{E}_\Delta$



Connections

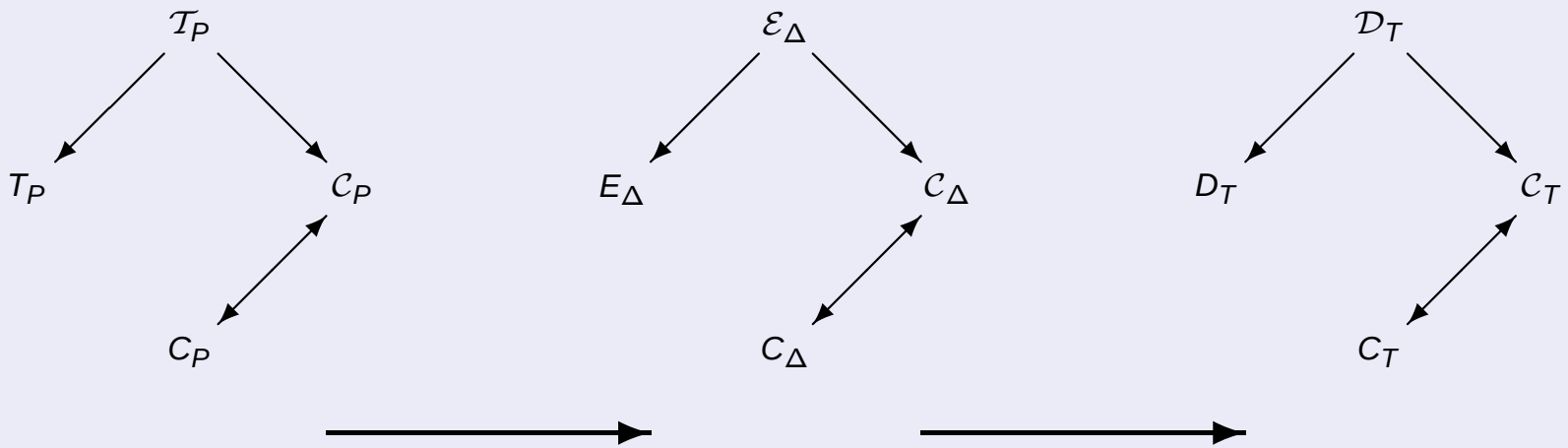
Strong parallels!



$c \leftarrow a, not\ b \quad \Rightarrow \quad \frac{a : \neg b}{c}$

Connections

Strong parallels!



$c \leftarrow a, \mathit{not}\ b \ \Rightarrow\ \frac{a : \neg b}{c}$ $\qquad$ $\frac{\alpha : \beta}{\gamma} \ \Rightarrow\ K\alpha \wedge \neg K \neg \beta \supset \gamma$

Computing with nonmon logics

Need programs with variables

- To facilitate modeling!
- General schema: answer-set programming
  - Encode problem constraints as finite programs
  - Represent problem instances as sets of ground atoms
  - So that extensions (expansions, stable models) of the union of the two represent solutions
- Most commonly used logic: logic programming with stable-model semantics

Computing with logic programs

Ground

- Instantiate all rules by replacing variables with constants in the program and data specification
- $ground(P)$ — grounding of P
- Allows to lift the semantics from the propositional case:
  - A set M of ground atoms (an Herbrand interpretation) is a stable model for P if M is a stable model for $ground(P)$

Solve

- Find stable models of the resulting ground (essentially, propositional) program

Computing with logic programs

Ground

- Instantiate all rules by replacing variables with constants in the program and data specification
- $ground(P)$ — grounding of P
- Allows to lift the semantics from the propositional case:
  - A set M of ground atoms (an Herbrand interpretation) is a stable model for P if M is a stable model for $ground(P)$

Solve

- Find stable models of the resulting ground (essentially, propositional) program

How to compute stable models?

Observations

- $ground(P)$ may be infinite; stable models may be infinite
- Thus, eliminate function symbols from the language
- Ensures that $ground(P)$ is finite and stable models are finite
- DATALOG$^{\neg}$

A precursor to all present implementations

- Two main modules
  - lparse — computes a subset of $ground(P)$ preserving stable models of P
  - smodels — computes stable models using an optimized version of a Davis-Putnam backtracking search procedure

Lparse

Grounds input programs

- Prtitions predicates into
  - domain predicates (no recursion through negation)
  - non-domain predicates (all others)
- Accepts domain-restricted programs
  - Rule is domain-restricted if: each variable appears in a domain-predicate atom that is non-negated in the body
  - Program is domain-restricted if each rule is

Domain predicates

Examples

- Facts — yes

  vtx(v). vtx(u). vtx(w). arc(v,u). arc(u,w).

- Non-recursive predicates — yes

  two-path(X,Y) :- arc(X,Z), arc(Z,Y), not arc(X,Y).

- Recursive predicates (no recursion through negation) — yes

  tc(X,Y) :- arc(X,Y).
  tc(X,Y) :- arc(X,Z), tc(Z,Y), vtx(Y).

- Recursion through negation — no

  in(X,Y) :- edge(X,Y), not out(X,Y).
  out(X,Y) :- edge(X,Y), not in(X,Y).

Domain-restriction

Why?

- Allows us to eliminate non-essential ground rules from $ground(P)$
- Subprogram of a program P consisting of rules defining domain predicates has a unique stable model M_{dom}
- M_{dom} is a subset of every stable model of the program
- $ground(P)$ contains for each rule all its ground instances
- Only those ground instances matter whose domain-predicate atoms in the body hold in M_{dom}
- M_{dom} can be computed quickly
  - possible optimizations using deductive database techniques

Role of domain-restriction

Example

- Given

  edge(1,2). clr(r). clr(b).
  :- clr(C), edge(X,Y), clrd(X,C), clrd(Y,C).

- The rule yields 64 (4^3) ground instances:

  ...
  :- clr(1), edge(r,2), clrd(r,1), clrd(r,1).
  ...

- Only 2 ground instances matter!

  :- clr(a), edge(1,2), clrd(1,a), clrd(2,a).
  :- clr(b), edge(1,2), clrd(1,b), clrd(2,b).

- Can be further simplified now

  :- clrd(1,a), clrd(2,a).
  :- clrd(1,b), clrd(2,b).

Domain-restriction

Allows limited support for function symbols

- Support for domain-restricted arithmetic

```
d(1). d(2). d(3).
even(X+1) :- d(X), X < 3, not even(X).
```

- Evaluate and simplify away during grounding

```
even(2) :- not even(1).
even(3) :- not even(2).
```

- Notation:

d(1..n). for d(1). d(2). ... d(n).

smodels

Computes stable models of programs produced by lparse

- Backtracking search for a set of atoms in $lparse(P)$ that is a stable model
- Similar to Davis-Putnam procedure for SAT
- Node in the search tree corresponds to a set of literals possibly consistent with a stable model
- To prune the search space:
  - propagation techniques (similar to unit propagation in SAT)
  - good heuristics to choose an atom for a choice point

Basic propagation rules in smodels

- Given a set of literals A:
  - derive a set of literals extending A and consistent with all stable models that are consistent with A
- Propagation does not eliminate stable models
- Examples:
  - If all literals in a rule are false except for exactly one, say L, derive L (unit propagation rule, as in SAT)
  - If there is only one rule c defining q and $q \in A$, derive all literals in the body of c
  - If all rules for q are blocked, derive *not* q
  - If no finite backward chain through positive atoms for q, then *not* q generalization of the well-founded model computation

Propagation — lookahead

- Assume *a* and apply basic propagation
  - if conflict, add *not* (*a*) to *A* and continue basic propagation
- Otherwise, assume *not* (*a*) and apply basic propagation
  - if conflict, add *a* to *A* and continue basic propagation
- Trade-off between time spent in propagation and its effect
- Currently — full lookahaed used in *smodels*

Extensions

- Additional syntax ("shorthands") to facilitate programming
- Extended semantics generalizing the stable-model semantics
- Choice rule: $\{a_1, \ldots, a_m\} : -L_1, \ldots, L_n.$
  - a_i — atoms; L_j — literals
  - If the body holds, then any subset of $\{a_1, ..., a_m\}$ (including empty) can be derived (is justified by the rule)
- Cardinality atom: $l\ \{L_1, \ldots, L_n\}\ u$
  - l, u — integer constants or variables
  - L_j — literals
  - satisfied in a model M if the number of the literals from $\{p_1, \ldots, p_n\}$ satisfied in M is between integers l and u (inclusive)
- Implicit representations of literals in cardinality atoms
  - $l\ \{p(X, Y) : d_1(X)\}\ u$ — d_1 is a domain predicate

Programming with smodels/lparse

Methodology

- Adding constraint : $-L_1, \ldots, L_n$ to program P eliminates all stable models of P that satisfy $L_1, \ldots, L_n$
- Leads to basic programming methodology: generate and test
  - Generator: provides candidate answer sets
    (typically encoded using even loops/choice rules)
  - Tester: eliminates those candidates that violate problem constraints
    (typically encoded using constraints)

Examples

Vertex cover (set of vertices “covering” all edges of a graph

```
%% vtx and edge specified as ground facts
%% k entered from the command line
{in(X)} :- vtx(X).
:- edge(X,Y), not in(X), not in(Y).
{in(X): vtx(X)} k.
```

Graph *k*-coloring

```
%% vtx and edge specified as ground facts
%% k entered from the command line
color(1..k).
1{clrd(X,C): color(C)}1 :- vtx(X).
:- edge(X,Y), clrd(X,C), clrd(Y,C).
```

Examples

Vertex cover (set of vertices “covering” all edges of a graph

```
%% vtx and edge specified as ground facts
%% k entered from the command line
{in(X)} :- vtx(X).
:- edge(X,Y), not in(X), not in(Y).
{in(X): vtx(X)} k.
```

Graph *k*-coloring

```
%% vtx and edge specified as ground facts
%% k entered from the command line
color(1..k).
1{clrd(X,C): color(C)}1 :- vtx(X).
:- edge(X,Y), clrd(X,C), clrd(Y,C).
```

Example — Hamiltonian path

Input data: directed graph, starting vertex

```
vtx(a). vtx(b). ...
edge(a,b). edge(c,a). ...
start(a).
```

Generator

```
{ hp(X,Y) } :- edge(X,Y).
```

Tester

```
%% Each vertex has at most one incoming edge and one outgoing edge
:-hp(X,Y), hp(X,Z), edge(X,Y), edge(X,Z), Y!=Z.
:-hp(Y,X), hp(Z,X), edge(Y,X), edge(Z,X), Y!=Z.
%% Every vertex is reachable from a given initial vertex
r(Y) :- start(Y).
r(Y) :- edge(X,Y), hp(X,Y), r(X).
:- vtx(X), not r(X).
```

Example — Hamiltonian path

Input data: directed graph, starting vertex

```
vtx(a). vtx(b). ...
edge(a,b). edge(c,a). ...
start(a).
```

Generator

```
{ hp(X,Y) } :- edge(X,Y).
```

Tester

```
%% Each vertex has at most one incoming edge and one outgoing edge
:-hp(X,Y), hp(X,Z), edge(X,Y), edge(X,Z), Y!=Z.
:-hp(Y,X), hp(Z,X), edge(Y,X), edge(Z,X), Y!=Z.
%% Every vertex is reachable from a given initial vertex
r(Y) :- start(Y).
r(Y) :- edge(X,Y), hp(X,Y), r(X).
:- vtx(X), not r(X).
```

Example — Hamiltonian path

Input data: directed graph, starting vertex

```
vtx(a). vtx(b). ...
edge(a,b). edge(c,a). ...
start(a).
```

Generator

```
{ hp(X,Y) } :- edge(X,Y).
```

Tester

```
%% Each vertex has at most one incoming edge and one outgoing edge
:-hp(X,Y), hp(X,Z), edge(X,Y), edge(X,Z), Y!=Z.
:-hp(Y,X), hp(Z,X), edge(Y,X), edge(Z,X), Y!=Z.
%% Every vertex is reachable from a given initial vertex
r(Y) :- start(Y).
r(Y) :- edge(X,Y), hp(X,Y), r(X).
:- vtx(X), not r(X).
```

Example — Hamiltonian path

Discussion

- Clauses

  ```
  :-hp(X,Y), hp(X,Z), edge(X,Y), edge(X,Z), Y!=Z.
  :-hp(Y,X), hp(Z,X), edge(Y,X), edge(Z,X), Y!=Z.
  ```

  can be replaced with:

  ```
  :- 2 { hp(X,Y):edge(X,Y) }, vtx(X).
  :- 2 { hp(X,Y):edge(X,Y) }, vtx(Y).
  ```

- The set of atoms $hp(x, y)$ in a stable model determines a Hamilton path starting in a
- All Hamilton paths starting in a are so specified
- lparse output has size that is linear in the size of input graph
- Hamilton-path problem is challenging for SAT solvers as no compact SAT encoding (with size linear in the size of input graph) is known

Other systems

Dlv

Leone, Eiter et al

- The same general approach as in smodels: ground and search
- Advanced grounding
  - dlv grounder borrows heavily from the area of deductive databases
  - rule rewriting and join optimization methods
- Disjunction in the heads
- Built-in arithmetic
- Support or aggregate operations (in particular, cardinality atoms)
- Propagation based on generalizations of well-founded semantics to the disjunctive case
- Expressive power: class Σ_P^2-search

Other solvers

There is much more

- Clasp (Schaub et al, Potsdam)
- Cmodels (Lierler, UT Austin)
- Assat (Lin and Zhao, Hong-Kong University of Science and Technology)

Thank you!