Stable Models Idea

- The construction of perfect models can be done without stratifying the program. Simply *guess* the model, process it into P and see if its least model coincides with the *guess*.
- If the program is stratified, the results coincide:
  - A correct *guess* must coincide on the 1st strata;
  - and on the 2nd (given the 1st), and on the 3rd …
- But this can be applied to non-stratified programs…

Stable Models Idea (cont)

- "Guessing a model" corresponds to "assuming default negations *not*". This type of reasoning is usual in NMR
  - Assume some default literals
  - Check in P the consequences of such assumptions
  - If the consequences completely corroborate the assumptions, they form a stable model
- The stable models semantics is defined as the intersection of all the stable models (i.e. what follows, no matter what *stable* assumptions)

SMs: preliminary example

$$\begin{array}{llll} a \leftarrow not\ b & c \leftarrow a & p \leftarrow not\ q & \\ b \leftarrow not\ a & c \leftarrow b & q \leftarrow not\ r & r \end{array}$$

Assume, e.g., *not r* and *not p* as true, and all other nots as false.
By processing this into P:

$$\begin{array}{llll} a \leftarrow \boldsymbol{false} & c \leftarrow a & p \leftarrow \boldsymbol{false} & \\ b \leftarrow \boldsymbol{false} & c \leftarrow b & q \leftarrow \boldsymbol{true} & r \end{array}$$

Its least model is *{not a, not b, not c, not p, q, r}*
So, it isn't a stable model:

- By assuming not r, r becomes true
- not a is not assumed and a becomes false

SMs example (cont)

$a \leftarrow not\ b$	$c \leftarrow a$	$p \leftarrow not\ q$	
$b \leftarrow not\ a$	$c \leftarrow b$	$q \leftarrow not\ r$	r

Now assume *not b* and *not q* as true, and all other nots as false.
By processing this into P:

$a \leftarrow$ ***true***	$c \leftarrow a$	$p \leftarrow$ ***true***	
$b \leftarrow$ ***false***	$c \leftarrow b$	$q \leftarrow$ ***false***	r

Its least model is *{a, not b, c, p, not q, r}*
I is a stable model
The other one is *{not a, b, c, p, not q, r}*
According to Stable Model Semantics:

c, *r* and *p* are true and *q* is false.
a and *b* are undefined

Stable Models definition

Let I be a (2-valued) interpretation of P. The definite program P/I is obtained from P by:

- deleting all rules whose body has *not A*, and $A \in I$
- deleting from the bodies all the remaining default literals

$$\Gamma_P(I) = \text{least}(P/I)$$

M is a stable model of P iff $M = \Gamma_P(M)$.

- *A* is true in P iff *A* belongs to all SMs of P
- *A* is false in P iff *A* doesn't belongs to any SMs of P (i.e. *not A* "belongs" to all SMs of P).

Properties of SMs

- ✓ Stable models are minimal models
- ✓ Stable models are supported
- ✓ If P is locally stratified then its single stable model is the perfect model
- ✓ Stable models semantics assign meaning to (some) non-stratified programs
  - E.g. the one in the example before

Importance of Stable Models

Stable Models were an important contribution:

- Introduced the notion of *default negation* (versus negation *as failure*)
- Allowed important connections to NMR. Started the area of LP&NMR
- Allowed for a better understanding of the use of LPs in Knowledge Representation

It is considered as THE semantics of LPs by a significant part of the community.

Default Logic

- To deal with incomplete information and default rules, Reiter introduced the *Default Logics*
- A theory Δ is a pair <T,D> where:
  - T is a set of 1st. order formulae (certain knowledge)
  - D is a set of default rules

Default Logic (Syntax)

- Default rules are of the form:

$$\frac{\phi : \psi}{\gamma}$$

  where ϕ, ψ and γ are formulae
  - ϕ are the pre-requisites
  - ψ are the justifications
  - γ are the conclusions
- Default rules with free variables are viewed as macros standing for their ground instatiations

Meaning of defaults

$$\frac{\phi : \psi}{\gamma}$$

- if ϕ is true, and it is consistent to assume ψ, then conclude γ
- Rules are to be maximally applied
- The semantics has also to make clear:
  - “true” where?
  - “consistent” with what?
  - how to add the conclusion?

Examples of default rules

$$\frac{\text{bird}(X) : \text{flies}(X)}{\text{flies}(X)}$$

$$\frac{\text{friend}(X,Y), \text{friend}(Y,Z) : \text{friend}(X,Z)}{\text{friend}(X,Z)}$$

$$\frac{\text{true} : \neg\text{connection}(X,Y)}{\neg\text{connection}(X,Y)}$$

$$\frac{\text{holds}(F,S) : \text{holds}(F,\text{res}(A,S))}{\text{holds}(F,\text{res}(A,S))}$$

$$\frac{\text{adult}(X) : \text{employed}(X), \neg\text{dropout}(X)}{\text{employed}(X)}$$

Default Logics (Semantics)

Let $\Delta = <T,D>$ be a theory, and E a set of literals. $\Gamma_\Delta(E)$ is the smallest set such that:

- it contains all logical consequences of T
- it is closed under rules ϕ/γ where $\phi:\psi/\gamma \in D$ and

$$T \cup E \not\models \neg\psi$$

E is a default extension of Δ iff

$$E = \Gamma_\Delta(E).$$

LP and Default Theories

Let Δ_P be the default theory obtained by transforming:

$$H \leftarrow B_1,\ldots,B_n, \text{not } C_1,\ldots, \text{not } C_m$$

into:

$$\frac{B_1,\ldots,B_n : \neg C_1,\ldots, \neg C_m}{H}$$

There is a one-to-one correspondence between the SMs of P and the default extensions of Δ_P

LPs as defaults

- LPs can be viewed as sets of default rules
- Default literals are the justification:
  - can be assumed if it is consistent to do so
  - are withdrawn if inconsistent
- In this reading of LPs, $\leftarrow$ is not viewed as implication. Instead, LP rules are viewed as inference rules.

Auto Epistemic Logic

- Adds a modal operator _ to denote knowledge
  - Allows expressing knowledge about the agent’s own knowledge
- Eg.
  - $\_A \wedge B \wedge \neg \_C \rightarrow D$ means “if I know A is true, B is true, and I don’t know whether C is true, then D is true”.

AEL and incomplete knowledge

- Allows (non-monotonic) completion of the knowledge:
  - concert → _concert

    “if there was a concert, I would know”. If I don’t have evidence for concert (i.e. I don’t know *concert*), then concert is false.
  - bird(X) ∧ ¬flies(X) → _¬flies(X)

    “I know all birds that do not fly”. Thus, if there is some bird, for which I have no evidence of non-flying, I conclude it doesn’t fly.

AEL (Moore’s Semantics)

A consistent theory T* is an expansion of the AEL theory T iff:

T* = T U {_F: T* |= F} U {¬_F: T* |≠ F}

- I know everything I can conclude from my theory.
- I don’t know everything that I cannot conclude from my theory (i.e. that doesn’t belong to all models of the theory)

LP and Auto-Epistemic Logic

Let T_P be the AEL theory obtained by transforming:

$$H \leftarrow B_1,\ldots,B_n, \text{not } C_1,\ldots, \text{not } C_m$$

into:

$$B_1 \wedge \ldots \wedge B_n \wedge \neg\_C_1 \wedge \ldots \wedge \neg\_C_m \rightarrow H$$

There is a one-to-one correspondence between the SMs of P and the (Moore) expansions of T_P

LPs as AEL theories

- LPs can be viewed as theories that refer to their own knowledge
- Default negation *not A* is interpreted as “*A* is not known”
- The LP rule symbol is here viewed as material implication

Extended LPs

- In Normal LPs all the negative information is implicit. Though that's desired in some cases (e.g. the database with flight connections), sometimes an explicit form of negation is needed for Knowledge Representation
- "Penguins **don't** fly" could be: $noFly(X) \leftarrow penguin(X)$
- This does not relate $fly(X)$ and $noFly(X)$ in:

$$fly(X) \leftarrow bird(X)$$
$$noFly(X) \leftarrow penguin(X)$$

For establishing such relations, and representing negative information a new form of negation is needed in LP:

Explicit negation - **∼**

Extended LP: motivation

- ~ is also needed in bodies:

  "Someone is guilty if is not innocent"

  - cannot be represented by: *guilty(X)* ← *not innocent(X)*
  - This would imply *guilty* in the absence of information about *innocent*
  - Instead, *guilty(X)* ← *~innocent(X)* only implies *guilty(X)* if *X* is proven not to be *innocent*
- The difference between *not p and ~p* is essential whenever the information about p cannot be assumed to be complete

ELP motivation (cont)

- ~ allows for greater expressivity:

  "If you're not sure that someone is not innocent, then further investigation is needed"

  - Can be represented by:

    $$investigate(X) \leftarrow not \sim innocent(X)$$

- ~ extends the relation of LP to other NMR formalisms. E.g
  - it can represent default rules with negative conclusions and pre-requisites, and positive justifications
  - it can represent normal default rules

ELP Language

- An Extended Logic Program P is a set of rules:

  $$L_0 \leftarrow L_1, \ldots, L_m, \textit{not } L_{m+1}, \ldots \textit{not } L_n \quad (n,m \geq 0)$$

  where the L_i are objective literals
- An objective literal is an atoms A or its explicit negation $\sim A$
- Literals *not* L_j are called *default literals*
- The Extended Herbrand base H_P is the set of all instantiated objective literals from program P
- We will consider programs as possibly infinite sets of instantiated rules.

ELP Interpretations

- An interpretation I of P is a set

  $$I = T \cup not\ F$$

  where T and F are disjoint subsets of H_P and

  $\sim L \in T \Rightarrow L \in F$ (Coherence Principle)

  i.e. if L is explicitly false, it must be assumed false by default
- I is total iff $H_P = T \cup F$
- I is consistent iff $\neg \exists$ L: $\{L, \sim L\} \subseteq T$
  - In total consistent interpretations the Coherence Principle is trivially satisfied

Answer sets

- It was the 1st semantics for ELPs [Gelfond&Lifschitz90]
- Generalizes stable models to ELPs

Let M^- be a stable models of the normal P^- obtained by replacing in the ELP P every ~A by a new atom A^-. An answer-set M of P is obtained by replacing A^- by ~A in M^-

A is true in an answer set M iff $A \in M$

A is false iff $\sim A \in M$

Otherwise, A is unknown

Some programs have no consistent answer sets:

e.g. $P = \{a \leftarrow, \sim a \leftarrow\}$

Answer sets and Defaults

Let Δ_P be the default theory obtained by transforming:

$$L_0 \leftarrow L_1,\ldots,L_m, \text{not } L_{m+1},\ldots, \text{not } L_n$$

into:

$$\frac{L_1,\ldots,L_m : \neg L_{m+1},\ldots, \neg L_n}{L_0}$$

where $\neg \sim A$ is (always) replaced by A

There is a one-to-one correspondence between the answer-sets of P and the default extensions of Δ_P

Answer-sets and AEL

Let T_P be the AEL theory obtained by transforming:

$$L_0 \leftarrow L_1,\ldots,L_m, \text{not } L_{m+1},\ldots, \text{not } L_n$$

into:

$$L_1 \wedge \_L_1 \wedge \ldots \wedge L_m \wedge \_L_m \wedge$$

$$\wedge \neg \_L_{m+1} \wedge \ldots \wedge \neg \_L_m \Rightarrow (L_0 \wedge \_L_0)$$

There is a one-to-one correspondence between the answer-sets of P and the expansions of T_P

WFS motivation

- Answer-sets (and stable models) are a good tool for representing knowledge. However:
  - its computation is NP-complete
  - it doesn't comply with various structural properties, desirable for goal-driven implementations
  - in various application domains, it is important to have efficient implementations for answering queries (that need not compute the whole model)
- The Well Founded Semantics is a weaker semantics
  - sound wrt stable models
  - with polynomial time complexity
  - amenable to goal-driven implementations

Cumulativity

A semantics *Sem* is **cumulative** iff for every P:

$\quad$ if $A \in Sem(P)$ and $B \in Sem(P)$ then $B \in Sem(P \cup \{A\})$

(i.e. all derived atoms can be added as facts, without changing the program's meaning)

This property is important for implementation:

without cumulativity, tabling methods cannot be used

Relevance

A directly depends on *B* if *B* occur in the body of some rule with head *A*. *A* depends on *B* if *A* directly depends on *B* or there is a *C* such that *A* directly depends on *C* and *C* depends on *B*.

A semantics *Sem* is **relevant** iff for every P:

$$A \in Sem(\mathrm{P}) \text{ iff } A \in Sem(\mathrm{Rel}_A(\mathrm{P}))$$

where $\mathrm{Rel}_A(\mathrm{P})$ contains all rules of P whose head is *A* or some *B* on which *A* depends on.

Only this property allows for the usual top-down execution of logic programs.

The logic of here-and-there

- As a basis for studying the language of answer sets and ist extensions, we look at the logic of two worlds: here and there

At each world w a set of atoms i(w) is verified

$$\begin{tikzcd} & \{T\} \\ \{H\} & t \\ h \arrow[ur] & \end{tikzcd}$$

$$i(h) = H$$
$$i(t) = T$$

Minimal models

- We define a partial ordering on models by

$\langle H,T\rangle \leq \langle H',T'\rangle \Leftrightarrow$
$T=T'$ & $H \subseteq H'$

- Given a theory **T**, a model $<H,T>$ of **T** is said to be **minimal** if it is minimal among models of **T** under the ordering ≤

Equilibrium logic

- A model <*H*,*T*> of a theory **T** is said to be an equilibrium model of **T** if it is a minimal model of **T** and *H*=*T*.
- Equilibrium logic is determined by the class of all equilibrium models of a theory, ie. a formula *a* is an equil. consequence of **T** iff *a* is true in all equil. models of **T**.

Some examples

- $\neg\neg a$ has no e. model
- $\neg a \text{ –> } b$ has a single e. model
- $\neg a \text{ —> } b$, $\neg b \text{ —> } a$ has 2 e. models

- consider $<\{\},\{a\}>$
- $<\{b\},\{b\}>$, for consider $<\{\},\{a\}>$
- $<\{a\},\{a\}>$, $<\{b\},\{b\}>$

Why negation-by-default?

- **J**-consequence: *p* is in $Cn_J(T)$ iff for all models M and worlds w, if M,w **|= T,** then M,w |= *p*.
- **J**-completions: E is a **J**-completion of T iff $E=Cn_J(\mathbf{T} \cup \{\neg a: a \notin E\})$
- Observation: equilibrium models correspond to **J**-completions
- Corollary: to reach equilibrium consistently add negated sentences until complete

Examples

- $\neg\neg p$ cannot be consistently completed by negation
- $\neg\neg p \rightarrow p$ can be completed by adding either $\neg p$ or $\neg\neg p$. In this case one of the corresponding e. models is “bigger” than the other.

“Minimal” equilibrium logic

- taking **J**-completions by negated **atoms** corresponds to selecting **minimal** equilibrium models (ie minimal in their verified atoms)

“Minimal” equilibrium logic

- defining **J**-completions using only the negation of **atoms** corresponds to choosing equilibrium models that are **minimal** (ie minimal in the set of verified atoms)

Equilibrium logic with strong negation

- based on here-and-there with strong negation(studied by Kracht, 1998). A 5-valued logic.
- equilibrium construction is the same, but at each world sets of literals(atoms and strongly negated atoms) are verified.
- Again, minimise true literals wrt (weakly) false literals. e. models are those whose literals are either true or weakly false.

Examples

- Let **T** = $\sim b$; $c \rightarrow b$; $\neg c \rightarrow d$; $\neg d \rightarrow c$.
- Let **T** = $\sim a$; $a \rightarrow c$; $b \rightarrow a$; $\neg b \rightarrow b$.
- Let **T** = $\sim b$; $\neg a \rightarrow b$

- Equilibrium model of **T** is $<\{\sim b,d\},\{\sim b,d\}>$
- **T** is inconsistent (no model)
- **T** has no e-model, by $<\{\sim b\},\{\sim b,a\}>$

Equilibrium logic with strong negation

- Based on here-and-there with strong negation(studied by Kracht, 1998). A 5-valued logic.
- Equilibrium construction is the same, but at each world sets of literals(atoms and strongly negated atoms) are verified.
- Again, minimise true literals wrt (weakly) false literals. e. models are those whose literals are either true or weakly false.

Syntactic Method : completion logics

- The idea: consider an intermediate logic L and a theory T in L. In palce of the L-consequences of T, form extensions E of T (called **completions**) that are complete in the sense that $\alpha \notin E \Rightarrow \neg\alpha \in E$. the logic L* is determined by the formulas true in all completions of T.

Syntactic Method : completion logics

- If L is an intermediate logic, T a theory in L. E is an **L-completion** of T iff
- $E = Cn_L(\mathbf{T} \cup \{\neg a : a \notin E\})$.
- Define a logic L* (in general nonmonotonic) such that $\alpha \in C_L{*}(T)$ if $\alpha \in E$ for each L-completion E of T.

Examples in J*

- $\neg\neg p$ cannot be completed consistently adding negated sentences
- $\neg\neg p$ –> p can be completed adding either $\neg p$ or $\neg\neg p$. Each determines a completion.

Observations

- The logic J* coincides con with the logic of equilibrium
- N5* coincides with N5-equilibrium
- Define J*min as J* with completions E such that $E=Cn_L(T\ U\ \{\neg a: a \notin E\})$ **for an atom *a*.**
- then J*min is the logic of equilibrium models that are **minimal** in the works ‘there’.

Stable model and answer set semantics

- Stable models first defined for **normal** logic programs(1988); ie for sets of formulas

a_1 & ..& a_n & $\neg b_1$ &...& $\neg b_m$ –> c, where a,b,c ´s are atoms.

- Answer sets generalise (1990) to **disjunctive** and **extended** programs, ie formulas

a_1 & ..& a_n & $\neg b_1$ &...& $\neg b_m$–>c_1 v..v c_k,

where a,b,c ´s are literals.

Some observations

- On normal, disjunctive and extended logic programs, stable models (resp. answer sets) correspond (exactly) to equilibrium models.
- The logic **J** of here-and-there is a maximal **deductive basis** for answer set inference, ie max monotonic sublogic in which equivalent theories have the same answer sets.

Further observations

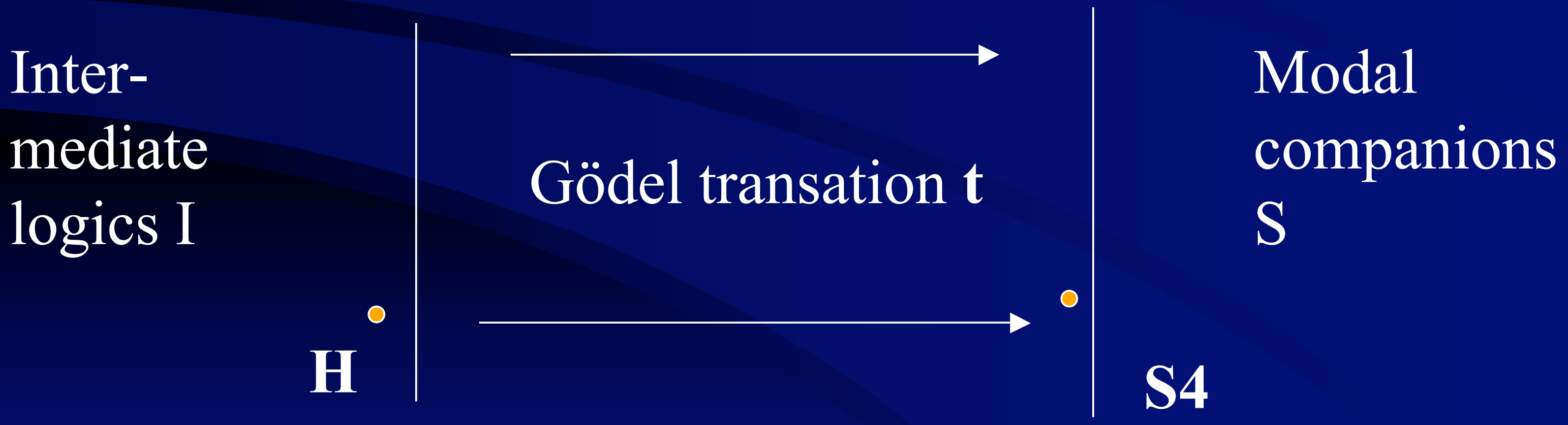
- Stable models of a program *P* correspond to the **J**-completions of *P*. But this can be extended:
- (1) it suffices to complete by negated atoms;
- (2) **J** can be replaced by intuitionistic logic **H**.
- (3) for extended programs replace **H** by **N**

Correspondences

- Each intermediate logic I has modal companions S under the Gödel (1933) translation *t*.
- In *t(p)* each subformula *a* of p is replaced by *La*.

Correspondences

$$\vdash_I p \quad \Leftrightarrow \quad \vdash_S t(p)$$



Correspondences

- Do these embeddings lift to the nonmonotonic case? Consider

$$E = Cn_J(P \cup \{\neg a \ : a \notin E\})$$

vs

$$E = Cn_S(P \cup \{\neg La \ : a \notin E\})$$

Correspondences

- If P is a logic program, then Gödel embedding extends, since completion is by negated **atoms**. So eg. stable models correspond to S4 expansions.

Programs with nested expressions Lifschitz, Tang, Turner, 1998

- Motivation: in addition to usual program rules, allow rules such as

  p <– ¬ (q, ¬r)   or p <– (q –> r; s) or even

  p; ¬q <– ¬~r
- Provide an adequate semantics that extends answer sets to such cases

Nested expressions

- Step 1: consider formulas of form *F –> G*, where *F,G* are any boolean combinations of atoms and strongly negated atoms (literals).
- Step 2: extend the concept of program **reduct** inductively to all boolean combinations of literals, and then to implications *(F –> G)*.
- Step 3: define answer set *S* in usual way, ie as minimal model of *P* reduced by *S*.

Some facts

- Any program is (answer set) equivalent to a program with formulas of form

a_1 & ..& a_n & ¬b_1 &...& ¬b_m–>c_1 v..v c_k v ¬d_1 v..v ¬ d_l

generalised answer sets correspond to equilibrium models

Some (LP-relevant) behaviour

- Representing conditional antecedents (bodies):-
- Can one re-write
- $(F \rightarrow G) \rightarrow H$ by $\neg (F \& \neg G) \rightarrow H$?
- Yes. If P is any program, then $P \cup (a \rightarrow b) \rightarrow c$ and $P \cup \neg (a \& \neg b) \rightarrow c$ have the same equilibrium models.
- But note that replacing $(F \rightarrow G)$ by $(\neg F \vee G)$ produces a different result.

Conservative extensions

- Introduce into a program *P* a new predicate by definition. Eg consider the program *P'* = *P* U {*F* –> *r*} where *F* is a formula in language of *P*, and *r* is a new predicate (atom).
- The extended program *P'* is a conservative extension of *P*. For any formula *G* in language of *P*, *P* |~ *G* iff *P'* |~ *G* .(Where |~ is equiibrium. consequence).

Stable vs partial stable inference

- Note that stable inference is ***supra-classical***,
- but classical logic is not a basis for stable inference...
- $\neg a \rightarrow b$ and $\neg b \rightarrow a$ are class. equivalent

- Likewise WF-inference (or inference on P-stable models) is ***supra-intuitionistic***
- but **H** is not a basis for WF-inference ....
- in **H**, ¬ a –> a derives ¬ a –> b.

P- stable models

- P-stable models can be represented as **J**-models,
- but do not appear to be characterisable (as minimal models) in the logic of here-and-there.
- Suggestion: change to (extensions of) **minimal** logic.

further observations

- Stable models of a program *P* correspond to the **J**-completions of *P*. But this can be extended:
- (1) it suffices to complete by negated atoms;
- (2) **J** can be replaced by intuitionistic logic **H**.
- (3) for extended programs replace **H** by **N**

Stable Models (corollary 1)

- Let P be a logic program (normal or disjunctive)
- M is a stable model of P iff
- $Th(M) = Cn_H(\mathbf{P} \cup \{\neg a : a \notin Th(M)\})$ (for atoms a), or
- $Th(M) = Cn_J(\mathbf{P} \cup \{\neg a : a \notin Th(M)\})$