

- we will consider now semantics for **negation-as-failure** in logic programming, which are in turn nonmonotonic systems
- a **normal logic program** is a set of rules of the form:

$$A \leftarrow B_1, \dots, B_n, \text{not } C_1, \dots, \text{not } C_n$$

- **predicate completion** introduced by R. Clark in 1978, was the first of several proposals in this sense
- it is based on the idea of translating normal programs to classical propositional logic
- let  $\Pi$  be a propositional normal program, and  $p$  a proposition in its language.  
The **definition** of  $p$  in  $\Pi$  is the set of rules in  $\Pi$  with the form

$$p \leftarrow L_1, \dots, L_n$$

being  $L_i = B_i$  or  $L_i = \text{not } C_i$

- let  $\{p \leftarrow L_{i,1}, \dots, L_{i,n_i}, 0 \leq i \leq m\}$  be the definition of an atom  $p$  in  $\Pi$ , then the **completion** of  $p$  is the predicate logic formula:

$$p \Leftrightarrow \bigvee_{0 \leq i \leq m} \left( \bigwedge_{0 \leq j \leq n_i} L_{i,j} \right)$$

- if  $m = 0$  the completion of  $p$  is  $p \Leftrightarrow \perp$
- let  $\Pi$  be a normal program, then the **completion** of  $\Pi$   $\text{COMPL}(\Pi)$  is the conjunction of all completion for every proposition in the language
- it is a propositional formula, interpreted under the **classic propositional semantics**
- with this semantics negation-as-failure *not* is interpreted as classical negation in a “different” theory.

There is no more the correspondence between  $\leftarrow$  in logic programming and  $\Rightarrow$  in propositional logic

- example:  $\Pi = \{p \leftarrow \text{not } q\}$
- example:  $\Pi = \{p \leftarrow \text{not } q, q \leftarrow \text{not } p\}$
- example:  $\Pi = \{p \leftarrow \text{not } p\}$

- example: let  $\Pi$  be the following normal program

$$\textit{bird}(\textit{tweety}) \leftarrow$$
$$\textit{bird}(X) \leftarrow \textit{penguin}(X)$$
$$\textit{fly}(X) \leftarrow \textit{bird}(X), \textit{not abnormal}(X)$$
$$\textit{abnormal}(X) \leftarrow \textit{penguin}(X)$$

Then  $\text{COMPL}(\Pi) \models \textit{fly}(\textit{tweety})$ , but  $\text{COMPL}(\Pi \cup \{\textit{penguin}(\textit{tweety})\}) \models \neg \textit{fly}(\textit{tweety})$

- **problems** with this semantics:
  - not every logic program has a consistent completion
  - for some programs it differs from negation-as-failure intuitions
  - some extensions are possible, but then we have an exponential blowup in the number of formulas in the completion
- in the FOL extension, terms are interpreted under the **UNA** and **DCA assumptions**.

- **stable models** semantics were introduced for logic programs by Gelfond and Lifschitz in 1988
- let  $\Pi$  be a normal logic program without negation-as-failure, and  $X$  a set of literals in the language. We say that  $X$  is **logically closed** if it is consistent, or it is the whole set of literals
- we say that  $X$  is **closed under  $\Pi$**  if for every rule  $A \leftarrow B_1, \dots, B_n$  in  $\Pi$  such that  $\{B_1, \dots, B_n\} \subseteq X$ , then  $A \in X$
- the set of **consequences** of  $\Pi$ , noted  $Cn(\Pi)$ , is the smallest set of literals in the language which is both logically closed and closed under  $\Pi$ .
- **Exercise:** prove that this set always exists for all  $\Pi$
- **Exercise:** prove that  $Cn(\Pi) = \text{lfp}(T_\Pi)$

- now we will extend this notion to normal programs with negation-as-failure
- let  $\Pi$  be a normal logic program, and  $X$  a set of literals in the language. We call the **reduct** of  $\Pi$  relative to  $X$  to the propositional logic program without negation as failure  $\Pi^X$  obtained by:
  - deleting from  $\Pi$  all rules of the form

$$A \leftarrow B_1, \dots, B_n, \text{not } C_1, \dots, \text{not } C_i, \dots, \text{not } C_m$$

with some  $C_i \in X$

- replacing each remaining rule

$$A \leftarrow B_1, \dots, B_n, \text{not } C_1, \dots, \text{not } C_m$$

with

$$A \leftarrow B_1, \dots, B_n$$

- since  $\Pi^X$  doesn't contain negation-as-failure,  $\mathbf{Cn}(\Pi^X)$  always exist
- we say that  $X$  is an **stable model** of  $\Pi$  iff  $X = \mathbf{Cn}(\Pi^X)$
- example:

$$p \leftarrow \text{not } q$$

$$p \leftarrow \text{not } r$$

$$q \leftarrow$$

- example:

$$p \leftarrow \text{not } p$$

- example:

$$p \leftarrow \text{not } q$$

- example:

$$p \leftarrow \text{not } q$$

$$q \leftarrow \text{not } p$$

- example:

$$p \leftarrow \text{not } q$$

$$q \leftarrow \text{not } p$$

$$r \leftarrow \text{not } r$$

$$r \leftarrow p$$

- **Theorem 34.** *Every logic program with negation-as-failure either has no stable model, or has at least one consistent stable model which is supported.*
- stable model also satisfy the **chain property**, if  $X \subset Y \subset Z$  and  $Y$  is an stable model of  $\Pi$ , then neither  $X$  nor  $Z$  are stable models of  $\Pi$
- **answer sets** are a generalization of stable models for logic programs with negation-as-failure, and strong negation
- stable models, like default logic, also can be used to define **credulous** and **skeptical** semantics

- introduced by van Gelder, Ross and Schlipf in 1990
- **well-founded** model is constructed with an extension of the consequence operator  $T_{\Pi}$  for programs without negation-as-failure
- let  $\Pi$  be a propositional normal program, and  $\mathcal{L}$  the set of all propositions in the language. Let  $\gamma_{\Pi} : 2^{\mathcal{L}} \longrightarrow 2^{\mathcal{L}}$  defined as follows:

$$\gamma_{\Pi}(X) = \mathbf{Cn}(\Pi^X)$$

- it is clear that any stable model of  $\Pi$  is a fixpoint of  $\gamma_{\Pi}$
- **Proposition 3.** *For any normal program  $\Pi$ ,  $\gamma_{\Pi}$  is anti-monotone.*
- **Exercise:** prove this proposition
- so  $\gamma_{\Pi}^2$  is monotone, and has least and greatest fixpoints

- if an atom belongs to the least fixpoint of  $\gamma_{\Pi}^2$ , it is called a **well-founded** atom in  $\Pi$ .  
if an atom doesn't belong to the greatest fixpoint of  $\gamma_{\Pi}^2$ , it is called an **unfounded** atom in  $\Pi$ .
- if we consider well-founded atoms as true, and unfounded atoms as false, then we obtain the **well-founded** three-valued model of  $\Pi$
- atoms that are not well-founded, nor unfounded are considered with truth value **undefined**
- the well-founded model exists, and is unique, for every normal program  $\Pi$

## Nonmonotonic Reasoning - the well-founded model

---

- example:

$$p \leftarrow \text{not } q$$

- example:

$$p \leftarrow \text{not } p$$

$$q \leftarrow$$

- example:

$$p \leftarrow \text{not } p$$

- example:

$$p \leftarrow \text{not } q$$

$$p \leftarrow \text{not } r$$

$$r \leftarrow \text{not } r$$

$$q \leftarrow$$

- example:

$$p \leftarrow \text{not } q$$

$$q \leftarrow \text{not } p$$

- **Proposition 4.** *The least and greatest fixpoints of  $\gamma_{\Pi}^2$  limits the fixpoints of  $\gamma_{\Pi}$  (stable models) from below and from above*
- **Exercise:** prove this proposition
- **Theorem 35.** *Any stable model of  $\Pi$  includes all well-founded literals, and no unfounded literal of  $\Pi$ .*
- **Theorem 36.** *The well-founded model of a normal program  $\Pi$  is two-valued iff it is the unique stable model of  $\Pi$*
- so the well-founded model is a **weaker** semantics for negation-as-failure than the stable models semantics

## Nonmonotonic Reasoning - the well-founded model

---

- the advantage of the well-founded model is that it is **always defined** for every normal program
- also, it has a better **computational complexity** than stable model semantics (answering a query under the well-founded model is **P**, while under the skeptical or credulous stable model semantics is **co - NP**)
- but it is **too skeptical** in comparison to the stable models. Even the skeptical stable models semantics for a normal program is more credulous than the well-founded model.

- the set of **observations**  $O$  is a set by propositions of the form

**f after**  $a_1, \dots, a_n$

where **f** is a fluent literal, and  $a_i$  are actions

- intuitively, the meaning of these propositions is that if  $a_1, \dots, a_n$  are executed from the initial situation then **f** holds in the state  $[a_n, \dots, a_1]$
- **special cases**:
  - if  $m = 0$  then

**initially f**

- **{initially f<sub>1</sub>, ..., initially f<sub>n</sub>}** is represented with como

**initially f<sub>1</sub>, ..., f<sub>n</sub>**