491 Knowledge Representation

# Minimal models and fixpoint semantics for definite logic programs

Marek Sergot
Department of Computing
Imperial College, London

January 2005 v1.0h

## Overview

A *definite logic program* is a set of clauses of the form

$$A \leftarrow A_1, \ldots, A_n \qquad (n \geq 0)$$

where $A$ and $A_1, \ldots, A_n$ are all *atoms*, i.e., *no negation.*

For a *definite* logic program $P$ (no negation), we have the following candidate semantics:

- the atomic (classical) consequences of $P$: $\{\alpha \mid \alpha \in \mathit{atoms}(P),\ P \models \alpha\}$
  i.e. $\mathit{atoms}(P) \cap \mathrm{Th}(P)$.
- the least (unique minimal) Herbrand model of $P$
- $\mathrm{lfp}(T_P)$ — the least fixpoint of $T_P$, the *immediate consequence operator* of $P$.
  $\mathrm{lfp}(T_P)$ is the least (set inclusion) set of atoms $I$ such that $T_P(I) = I$.
- $T_P {\uparrow}^\omega$
- $T'_P {\uparrow}^\omega(\emptyset)$ where $T'_P(I) \stackrel{\text{def}}{=} T_P(I) \cup I$

For definite clause programs these different semantics are *all equivalent.*

Moreover, in deductive databases it is usual to split a logic program $P$ into $\mathit{IDB}$ (rules) and $\mathit{EDB}$ (facts)

$$P = \mathit{IDB} \cup \mathit{EDB}$$

Then we have also the following semantics for $P$

- $\mathit{Cl}_{\mathit{IDB}}(\mathit{EDB})$ — the closure of $\mathit{EDB}$ under the rules $\mathit{IDB}$ — the smallest set of atoms $X$ such that
  - $\mathit{EDB} \subseteq X$
  - $T_{\mathit{IDB}}(X) \subseteq X$
- $T'_{\mathit{IDB}} {\uparrow}^\omega(\mathit{EDB})$

Most of this does not hold for *normal* logic programs, i.e., where there can be occurrences of negation-by-failure in the body of a clause. We will look at ways to modify it for normal logic programs later.

## Preliminary basics: minimal models

Given a set $P$ of first-order formulas (a theory, a program, a database, or a knowledge base) an *interpretation* is constructed thus:

(1) Choose a domain $D$.

(2) Specify a mapping from ground terms of $P$ to the elements of $D$ which says what each ground term denotes in your interpretation.

Steps (1) and (2) together specify a *pre-interpretation.* Now ...

(3) Give the extension of every predicate of $P$ in the interpretation, by saying which ground atoms of $P$ are true and which are false.

A *model* of $P$ is an interpretation of $P$ in which every formula of $P$ is *true.*

The (well-formed formula) $\alpha$ is a *logical consequence* of $P$

$$P \models \alpha$$

when $\alpha$ is true in *every* model of $P$. In the earlier notation this is $\alpha \in \mathrm{Th}(P)$.

## Herbrand models

A *Herbrand interpretation* is a very special kind of interpretation: ground terms denote themselves.

- The domain $D$ is the set of all ground terms in the language of $P$ (the *Herbrand universe* $\mathrm{U}_P$).
- Each ground term of $P$ denotes itself.
- It only remains to say which ground atoms of $P$ are true and which are false.

So, to specify a Herbrand interpretation for a program or database $P$, only step (3) is required – the pre-interpretation steps (1) and (2) are fixed.

A *Herbrand model* is a Herbrand interpretation which is a model.

Why do Herbrand interpretations have special status?
Because – in certain circumstances – there is no need to look at *all* possible interpretations and models; it is sufficient to look at Herbrand interpretations and models only. This obviously simplifies everything.
What are these circumstances?

**Theorem** If $P$ is a set of universal sentences (all quantifiers are 'at the front' and all are universal) then:

$P$ has a model if and only if $P$ has a Herbrand model

*Proof*: One half is trivial: if $P$ has a Herbrand model then obviously $P$ has a model. For the other half, given a model $\mathcal{M}$ of $P$, it is very easy to show how to construct a Herbrand model for $P$ as well.

*Practical significance* Suppose we have:

- a program or database $P$ of universal sentences (say clauses);
- a query $\leftarrow G$ (which is a universal sentence because this is just conventional shorthand for $\forall(\neg G)$).

Now

$$\begin{array}{lll} P \models \exists(G) & \text{iff} & P \cup \{\leftarrow G\} \text{ has no model} \\ & \text{iff} & P \cup \{\leftarrow G\} \text{ has no Herbrand model.} \end{array}$$

Very convenient!

## How to specify Herbrand interpretation

In order to specify a Herbrand interpretation it is enough to say – step (3) – which ground atoms are true and which are false (because the pre-intepretation steps (1) and (2) are fixed for Herbrand interpretations).

The set of ground *atoms* of the language of $P$, denoted

$$atoms(P) \quad \text{or sometimes } \mathrm{B}_P$$

is called the *Herbrand base* of $P$.

So to specify a Herbrand interpretation for a program or database $P$ it is enough to say, for each $\alpha$ in $atoms(P)$, whether $\alpha$ is true or false.

**Example** Database $P$:
$$\begin{array}{l} p(a) \leftarrow \\ p(b) \leftarrow \\ q(X) \leftarrow \ p(X) \end{array}$$

The Herbrand base $atoms(P)$ is $\{p(a), p(b), q(a), q(b)\}$.

Here is a Herbrand interpretation of $P$:

$$\{p(a) \mapsto \text{true}, p(b) \mapsto \text{false}, q(a) \mapsto \text{false}, q(b) \mapsto \text{true}\}.$$

This is an interpretation, but not a model of $P$.

Here is another Herbrand interpretation of $P$:

$$\{p(a) \mapsto \text{true}, p(b) \mapsto \text{true}, q(a) \mapsto \text{true}, q(b) \mapsto \text{true}\}.$$

This *is* a model of $P$.

## Herbrand interpretations as sets of atoms

Listing out, for each atom $\alpha$ in $atoms(P)$, whether it is true or false is tedious – and unnecessary. We can adopt the following convention instead.

Given a database or program $P$, $atoms(P)$ is fixed. To specify a Herbrand interpretation, it is sufficient to say which atoms of $P$ are *true*. The remainder are false by convention.
So every Herbrand interpretation of $P$ can be identified with a subset of $atoms(P)$. These are the ground atoms that are true in the interpretation.
Note that we could choose the other convention, and list out all the false atoms instead. But this is not so convenient.

**Example** Database $P$:
$$\begin{array}{l} p(a) \leftarrow \\ p(b) \leftarrow \\ q(X) \leftarrow \ p(X) \end{array}$$

Here are some Herbrand interpretations of $P$:

| | |
|---|---|
| $\{p(a), q(b)\}$ | This is *not* a model of $P$ |
| $\{p(a), p(b), q(a), q(b)\} = atoms(P)$ | This *is* a model of $P$ |
| $\emptyset$ (the empty set) | This is *not* a model of $P$ |

(There are other Herbrand models of $P$.)

**Note** It is important not to confuse two different things. Associating Herbrand models with subsets of $atoms(P)$ is *nothing to do* with 'Closed World Assumptions'. It is just a convenient and concise way of representing Herbrand interpretations of $P$.

**Example** Database $P$:

$$p \leftarrow q, r$$

Here are some Herbrand interpretations of $P$:

| | |
|---|---|
| $\{q, r\}$ | This is *not* a model of $P$ |
| $\{p, q, r\} = atoms(P)$ | This *is* a model of $P$ |
| $\emptyset$ | This *is* a model of $P$ |

(There are other Herbrand models of $P$.)

**Final remark** In all my examples so far, the Herbrand base $atoms(P)$ is finite. This saves writing for me. But $atoms(P)$ is not always finite. $atoms(P)$ is not finite if there are function symbols in $P$.

## Minimal (Herbrand) models

The relation $\subseteq$ (subset) is obviously a partial ordering for subsets of $atoms(P)$, and therefore for Herbrand interpretations and Herbrand models. So we can talk of *minimal* Herbrand models.

**Definition** A Herbrand model is *minimal* if no proper subset of it is also a model.
Note that *minimal* does not imply *unique*, in general. A set of formulas might have several minimal Herbrand models. For example, $\{a \vee b\}$ has the following Herbrand models:

$$\{a, b\}$$

$$\{a\} \qquad\qquad \{b\}$$

The models $\{a\}$ and $\{b\}$ are both minimal.

The usual terminology for any partial ordering is that *least* means *unique* minimal. So sometimes we speak of the *least* Herbrand model, which is the unique minimal model, if it exists.

## Definite clauses

Sets of *definite clauses* (atom in the head, no negations anywhere) have some very convenient and very important properties.

**Theorem** Let $P$ be a set of definite clauses. Then:

- $atoms(P)$ is always a model of $P$;
- *Model intersection property*
  If $M_1$ is a model of $P$ and $M_2$ is a model of $P$ then $M_1 \cap M_2$ is a model of $P$.

**Theorem** Let $P$ be a set of definite clauses. Then:

- $P$ has a model. ($atoms(P)$ is one.)
- $P$ has a minimal Herbrand model.
- $P$ has a *unique* minimal Herbrand model, denoted $\mathbf{M}(P)$.
- $\mathbf{M}(P)$ = the intersection of all Herbrand models of $P$.

These very strong properties do not hold in general. They hold for sets of *definite* clauses.

A natural contender for the intended semantics of a definite clause program $P$ is its least (unique minimal) Herbrand model $\mathbf{M}(P)$.

How does this new semantics relate to what we had earlier? They are equivalent!

**Theorem** Let $P$ be a set of definite clauses. The minimal Herbrand model $\mathbf{M}(P)$ is the set of all ground atoms of $P$ that are logical consequences of $P$:

$$\mathbf{M}(P) = \{\alpha \mid \alpha \in atoms(P) \text{ and } P \models \alpha\} = atoms(P) \cap \mathrm{Th}(P) \text{ in the other notation}$$

*Proof* $\alpha \in atoms(P)$ and $P \models \alpha$
iff $\alpha \in atoms(P)$ and $P \cup \{\leftarrow \alpha\}$ has no model
iff $\alpha \in atoms(P)$ and $P \cup \{\leftarrow \alpha\}$ has no Herbrand model
iff $\alpha \in atoms(P)$ and $\neg\alpha$ is false in all Herbrand models of $P$
iff $\alpha \in atoms(P)$ and $\alpha$ is true in all Herbrand models of $P$
iff $\alpha \in atoms(P)$ and $\alpha$ is in the intersection of all Herbrand models of $P$
iff $\alpha \in atoms(P)$ and $\alpha \in \mathbf{M}(P)$.

## 'Supported' interpretations

The notion of a 'supported' interpretation of a set of clauses is not so important for definite clause programs but it will come up later so I might as well deal with it now.

*Motivating example* Consider the program or database $P$:

$p(a) \leftarrow$
$q(b) \leftarrow$
$r(X) \leftarrow\ p(X)$

Here is a Herbrand model of $P$:

$$M_1 = \{p(a), q(a), q(b), r(a), r(b)\} \quad \text{(not minimal)}$$

And here is another:

$$\mathbf{M}(P) = \{p(a), q(b), r(a)\} \quad \text{(minimal)}$$

Apart from minimality, is there anything else to distinguish model $\mathbf{M}(P)$ from model $M_1$?

Notice that in $\mathbf{M}(P)$, every atom is the head of a clause of $P$ whose body is true in $\mathbf{M}(P)$. We say that such a model (or interpretation more generally) is *supported*. This is not the case in $M_1$. In $M_1$ there are atoms ($q(a)$ and $r(b)$) which have no 'support' in this sense: they are not the heads of any clause in $P$ whose body is true in $M_1$.

**Definition** Let $I$ be a Herbrand interpretation of a set $P$ of clauses (not necessarily definite). $I$ is a *supported* interpretation of $P$ if and only if, for every $\alpha \in I$:

$$\alpha \leftarrow B_1, \ldots, B_m \qquad (m \geq 0)$$

is a ground instance of a clause in $P$, and $B_1, \ldots, B_m$ are all true in $I$, i.e., $\{B_1, \ldots, B_m\} \subseteq I$.

Notice that 'supported' is defined for *interpretations* not just models. An interpretation of $P$ can be a supported interpretation without being a model of $P$. For example, same example $P$ as above:

$$\begin{array}{l} p(a) \leftarrow \\ q(b) \leftarrow \\ r(X) \leftarrow\ p(X) \end{array}$$

The interpretation $\{p(a), r(a)\}$ is supported, but is not a model.

When we look at normal logic programs and extended logic programs we shall be interested in their (not necessarily unique) *minimal supported* Herbrand models. For definite clause programs, 'supported' is not so interesting: the minimal Herbrand model $\mathbf{M}(P)$ is unique, and it's easy to see it is always supported as well.

## Fixpoint semantics

Again, the following holds for *definite* clause programs. We will look at how it generalises for normal logic programs and extended logic programs later.

### The immediate consequence operator $T_P$

Every definite clause program or database $P$ has an operator $T_P$ – the *immediate consequence* operator – associated with it.

$T_P$ maps subsets of *atoms*$(P)$ to subsets of *atoms*$(P)$:

$$T_P\colon \wp(\mathit{atoms}(P)) \to \wp(\mathit{atoms}(P))$$

Or if you prefer, $T_P$ maps sets of ground atoms of $P$ to sets of ground atoms of $P$, or Herbrand intrepretations of $P$ to Herbrand interpretations of $P$.

**Definition** Let $P$ be a set of definite clauses and $I$ a set of ground atoms of $P$.

$$\begin{aligned} T_P(I) = \{\alpha \in \mathit{atoms}(P) \mid & \alpha \leftarrow B_1, \ldots, B_m\ (m \geq 0) \text{ is a ground instance of a clause in } P \\ & \text{and } \{B_1, \ldots, B_m\} \subseteq I\} \end{aligned}$$

Or in words: $\alpha \in T_P(I)$ iff $\alpha \leftarrow B_1, \ldots, B_m (m \geq 0)$ is a ground instance of a clause in $P$ and the conditions $B_1, \ldots, B_m$ of the body are all true in the interpretation $I$.

It should be obvious why $T_P$ is called the 'immediate consequence' operator.

**Example** The program or database $P$:

$$\begin{array}{l} p(a) \leftarrow \\ q(b) \leftarrow \\ r(X) \leftarrow\ p(X) \end{array}$$

$$\begin{aligned} T_P(\emptyset) &= \{p(a), q(b)\} \\ T_P(\{p(a), q(b)\}) &= \{p(a), q(b), r(a)\} \\ T_P(\{p(a), q(b), r(b)\}) &= \{p(a), q(b), r(a)\} \\ T_P(\{q(a)\}) &= \{p(a), q(b)\} \\ T_P(\{p(b)\}) &= \{p(a), q(b), r(b)\} \\ &\vdots \end{aligned}$$

For *definite clause* programs $P$, the immediate consequence operator $T_P$ has some very important properties:

1. $T_P$ is monotonic:
$$\text{If } I_1 \subseteq I_2 \text{ then } T_P(I_1) \subseteq T_P(I_2)$$
   This property is actually implied by the next.
2. $T_P$ is 'continuous'.

For present purposes, it does not matter what 'continuous' means. The definition takes too long to state. I'll give a more careful account later. What matters is that there are some important results which hold for continuous operators.

In particular:

- a continuous operator $T_P$ has a *fixpoint* $I$
$$T_P(I) = I$$
- it has a *least* (unique minimal) fixpoint
- the least fixpoint is
$$T_P{\uparrow}^{\omega} \qquad \text{(defined in a moment)}$$

## The significance of fixpoints of $T_P$

**Theorem** The interpretation (set of ground atoms) $I$ is a *model* of $P$ if and only if

$$T_P(I) \subseteq I \qquad \text{(‘$I$ is closed under $T_P$’)}$$

*Proof* Trivial: just apply the definition of $T_P$.

**Theorem** The interpretation (set of ground atoms) $I$ of $P$ is *supported* if and only if

$$T_P(I) \supseteq I$$

*Proof* Trivial: just apply the definition of $T_P$.

So now any *fixpoint* $I$ of $T_P$

$$T_P(I) = I$$

will be a *supported model* of $P$. And the *least* such fixpoint $T_P{\uparrow}^\omega$ will be the (unique) *minimal supported model* of $P$. This $I$ is a natural candidate for the semantics of $P$.

## The least fixpoint of $T_P$

There is a standard result for all ‘continuous’ operators such as $T_P$ (which is ‘continuous’ when $P$ is a set of definite clauses). This result says

$$\mathrm{lfp}(T_P) = T_P{\uparrow}^\omega$$

(lfp is ‘least fixpoint’).

**Definition**

$$\begin{aligned}
T_P{\uparrow}^0 &\overset{\text{def}}{=} \emptyset \\
T_P{\uparrow}^1 &\overset{\text{def}}{=} T_P(T_P{\uparrow}^0) \qquad (= T_P(\emptyset)) \\
T_P{\uparrow}^2 &\overset{\text{def}}{=} T_P(T_P{\uparrow}^1) \\
&\vdots \\
T_P{\uparrow}^{n+1} &\overset{\text{def}}{=} T_P(T_P{\uparrow}^n) \\
&\vdots \\
T_P{\uparrow}^\omega &\overset{\text{def}}{=} \textstyle\bigcup_{n\geq 0} T_P{\uparrow}^n
\end{aligned}$$

Now the key result:

**Theorem** (van Emden and Kowalski)
If $P$ is a set of definite clauses, its unique minimal Herbrand model $\mathbf{M}(P)$ is given by

$$\mathbf{M}(P) = \mathrm{lfp}(T_P) = T_P{\uparrow}^\omega$$

*Proof* The first half of the equation is quite easy to prove. The second half just uses the standard result once it is established that $T_P$ is continuous.

**Example** Suppose $P$ is:

$$\begin{aligned}
&q(a,b) \leftarrow \\
&p(b) \leftarrow \\
&p(X) \leftarrow\ q(X,Y), p(Y)
\end{aligned}$$

The minimal Herbrand model $\mathbf{M}(P)$ is obviously

$$\mathbf{M}(P) = \{q(a,b), p(a), p(b)\}$$

by inspection.

The least fixpoint $\mathrm{lfp}(T_P)$ is obtained thus:

$$\begin{aligned}
&T_P{\uparrow}^0 = \emptyset \\
&T_P{\uparrow}^1 = T_P(\emptyset) = \{q(a,b), p(b)\} \\
&T_P{\uparrow}^2 = T_P(\{q(a,b), p(b)\}) = \{q(a,b), p(b), p(a)\} \\
&T_P{\uparrow}^3 = T_P(\{q(a,b), p(b), p(a)\}) = \{q(a,b), p(b), p(a)\} \\
&\quad\vdots \\
&T_P{\uparrow}^\omega = \{q(a,b), p(b), p(a)\} = \mathbf{M}(P)
\end{aligned}$$

$\mathbf{M}(P)$ is a model of $P$: $\quad T_P(\mathbf{M}(P)) \subseteq \mathbf{M}(P)$
$\mathbf{M}(P)$ is supported: $\quad T_P(\mathbf{M}(P)) \supseteq \mathbf{M}(P)$

**Example** Consider the database/program $P = \{p \leftarrow q,\ q \leftarrow p\}$. Its Herbrand interpretations are:

$$\{p,q\}$$

$$\{p\} \qquad\qquad \{q\}$$

$$\{\}$$

| | | | |
|---|---|---|---|
| $T_P(\{p,q\})$ | $=$ | $\{p,q\}$ | fixpoint – supported model; but not minimal |
| $T_P(\{p\})$ | $=$ | $\{q\}$ | not a model; not supported |
| $T_P(\{q\})$ | $=$ | $\{p\}$ | not a model; not supported |
| $T_P(\{\})$ | $=$ | $\{\}$ | fixpoint – supported model; minimal |

**Example** Consider the database/program $P = \{q(0) \leftarrow,\; p(X) \leftarrow p(f(X))\}$.

Is $\{q(0), p(f(f(0))), p(f(0)), p(0)\}$ a model of $P$? Yes:

$$T_P(\{q(0), p(f(f(0))), p(f(0)), p(0)\}) = \{q(0), p(f(0)), p(0)\}$$

Is $\{q(0), p(0)\}$ a model of $P$? Yes:

$$T_P(\{q(0), p(0)\}) = \{q(0)\} \subseteq \{q(0), p(0)\}$$

Is $\{q(0)\}$ a model of $P$? Yes:

$$T_P(\{q(0)\}) = \{q(0)\} \qquad \text{(So also supported, and fixpoint)}$$

Is $\emptyset$ a model of $P$? No:

$$T_P(\emptyset) = \{q(0)\}$$

## Deductive databases

The simplest form of deductive database:

| | | |
|---|---|---|
| *'rules'* | IDB | *definite clauses* |
| *'facts'* | EDB | *atoms* |

EDB is the 'extensional database'
IDB is the 'intensional database'
(stupid terminology)

The database content is

$$Cl_{IDB}(EDB)$$

the 'closure of facts $EDB$ under rules $IDB$'

**Definition** $Cl_{IDB}(EDB)$ is the *smallest* (set inclusion) set $X$ of atoms such that:

(1) $EDB \subseteq X$

(2) $T_{IDB}(X) \subseteq X$ ('$X$ is closed under $IDB$')

How do we know $Cl_{IDB}(EDB)$ exists and is unique?

- $atoms(IDB \cup EDB)$, the set of all ground atoms of $IDB \cup EDB$, satisfies conditions (1) and (2) above:
  - $EDB \subseteq atoms(IDB \cup EDB)$ ($EDB$ is a set of ground atoms)
  - $T_{IDB}(atoms(IDB \cup EDB)) \subseteq atoms(IDB \cup EDB)$
- If $T_{IDB}$ is monotonic (which it is when the rules $IDB$ are all definite clauses) then:

  if $X_1$ satisfies conditions (1) and (2) above, and $X_2$ satisfies conditions (1) and (2), then $X_1 \cap X_2$ satisfies conditions (1) and (2). (Easy exercise.)

  So $Cl_{IDB}(EDB)$ is the intersection of all sets of ground atoms that satisfy conditions (1) and (2).

How do we compute $Cl_{IDB}(EDB)$ ?

Let

$$T'_{IDB}(I) \overset{\text{def}}{=} T_{IDB}(I) \cup I$$

Then:

$$Cl_{IDB}(EDB) = T'_{IDB}{\uparrow}^{\omega}(EDB)$$

**Exercise** $IDB \cup EDB$ is also a set of definite clauses. Show:

$$T_{IDB \cup EDB}{\uparrow}^{\omega} = T'_{IDB}{\uparrow}^{\omega}(EDB)$$

**Exercise** Construct an example (see Tutorial Exercise sheet) to show that, in general

$$T_{IDB}{\uparrow}^{\omega}(EDB) \neq T'_{IDB}{\uparrow}^{\omega}(EDB)$$

This is the basis of

- *bottom-up* computation methods for deductive databases

The *'naive'* method implements $T'_{IDB}{\uparrow}^{\omega}(EDB)$ directly.

The *'semi-naive'* method uses the fact that

- $X \subseteq T'_{IDB}(X)$

to eliminate unnecessary recomputation on each iteration.
(There are other methods (e.g. *'not so naive'*) which are a bit cleverer.)

## Normal logic programs

The above all holds for *definite clauses*. If we allow negative conditions in clauses ('normal logic programs') then none of the nice properties hold in general.

**Definition** Let $P$ be a normal logic program and $I$ a set of ground atoms of $P$.

$$\begin{aligned} T_P(I) = \{\alpha \in \mathit{atoms}(P) \mid\ & \alpha \leftarrow L_1, \ldots, L_m\ (m \geq 0) \text{ is a ground instance of a clause in } P \\ & \text{and for all } L_i,\ 0 \leq i \leq m\text{:} \\ & \text{if } L_i \text{ is an atom, then } L_i \in I\text{; if } L_i \text{ is of the form not } B_i\text{, then } B_i \notin I\} \end{aligned}$$

Equivalently: if we write $\mathit{head}(r)$ for the head of a clause $r$, $\mathit{body}^+(r)$ for the set of positive atoms in the body of $r$, and $\mathit{body}^-(r)$ for the set of atoms appearing in negation-by-failure literals in the body of $r$, then:

$$\begin{aligned} T_P(I) = \{\mathit{head}(r) \mid\ & r \text{ is a ground instance of a clause in } P, \\ & \mathit{body}^+(r) \subseteq I,\ \text{and } \mathit{body}^-(r) \cap I = \emptyset\ \} \end{aligned}$$

Let $I$ be a set of ground atoms of a normal logic program $P$.

- $I$ is a *model* of $P$ iff $T_P(I) \subseteq I$ ('$I$ is closed under $T_P$')
- $I$ is a *supported interpretation* of $P$ iff $T_P(I) \supseteq I$

So we are still interested in fixpoints $T_P(I) = I$, and in minimal fixpoints. But there is no guarantee, in general, that a fixpoint exists, or if it exists, that there is a unique minimal fixpoint.

For example, consider the logic program/database $P$:

$p(1) \leftarrow$
$q(2) \leftarrow$
$r(X) \leftarrow \text{not } q(X)$

- There are two *different* minimal models of $P$:

  $$\{p(1), q(2), r(1)\} \quad \text{and} \quad \{p(1), q(2), q(1)\}$$

  Note that the first one is supported and the second one is not. (The atom $q(1)$ in the second model has no support.)

- The model intersection property does not hold. For example:

  $$\{p(1), q(2), r(1)\} \cap \{p(1), q(2), q(1)\} = \{p(1), q(2)\}$$

  is not a model of $P$.

- The immediate consequence operator $T_P$ is no longer continuous. It is not even monotonic, in general. For example, for the $P$ above:

  $$\begin{aligned} T_P(\emptyset) &= \{p(1), q(2), r(1), r(2)\} \\ T_P(\{q(2)\}) &= \{p(1), q(2), r(1)\} \end{aligned}$$

  Clearly $\emptyset \subseteq \{q(2)\}$ but $T_P(\emptyset) \not\subseteq T_P(\{q(2)\})$

**Example** Even simpler: consider $P = \{p \leftarrow \text{not } q\}$.

- $T_P$ is not monotonic: $T_P(\emptyset) = \{p\}$ but $T_P(\{q\}) = \emptyset$.
- $\{p\}$ and $\{q\}$ are both (minimal) models:

  $T_P(\{p\}) = \{p\}$ (and so a fixpoint, and supported);
  $T_P(\{q\}) = \emptyset$ (a model, but not supported)

**Example** Consider $P = \{p \leftarrow \text{not } q,\ q \leftarrow \text{not } p\}$.
$\{p\}$ and $\{q\}$ are both (minimal) supported models:

$$T_P(\{p\}) = \{p\} \quad \text{and} \quad T_P(\{q\}) = \{q\}$$

Let's try to compute $T_P{\uparrow}^\omega$:

$$\begin{aligned} T_P(\emptyset) &= \{p, q\} \\ T_P(\{p, q\}) &= \emptyset \end{aligned}$$

Not getting anywhere!

Note that $T_P{\uparrow}^\omega = \bigcup_{n \geq 0} T_P{\uparrow}^n = \emptyset \cup \{p, q\} = \{p, q\}$.
But $T_P(\{p, q\}) \neq \{p, q\}$. So $T_P{\uparrow}^\omega$ is not a fixpoint of this $T_P$.

Note, in this case, $\{p, q\}$ is a model of $P$, though obviously not a minimal model. Also not supported.

## Small remark

The references to $T_P{\uparrow}^\omega$ — as opposed to $T_P{\uparrow}^\omega(\emptyset)$, which is the same thing — and to the property that $T_P$ is 'continuous' are for historical reasons only. That is the way the van Emden-Kowalski theorem was originally stated and proved.

For our purposes it does not matter what 'continuous' means. Here follows all the technical background that we need.

## Operators, closures

We will keep seeing many of the same properties over and over again. This next bit of material will come later but you might find it helpful to see some of it now. If you don't find it helpful, ignore it for now.

Here is a summary of the theory of operators in a set $U$. The first part is very general. $U$ can be any set. Operators are just mappings of $\wp(U)$ into itself.

**Definition**

1. An *operator* in $U$ is any mapping $F\colon \wp(U) \to \wp(U)$.
2. An operator $F$ is called *monotonic* (sometimes: monotone) if, for all subsets $S_1$, $S_2$ of $U$, $S_1 \subseteq S_2 \;\Rightarrow\; F(S_1) \subseteq F(S_2)$.
3. An operator $F$ is called *anti-monotonic* (sometimes: anti-monotone) if, for all subsets $S_1$, $S_2$ of $U$, $S_1 \subseteq S_2 \;\Rightarrow\; F(S_2) \subseteq F(S_1)$.
4. An operator $F$ is called *compact* (sometimes: finitizable) if, for all subsets $S$ of $U$, if $\alpha \in F(S)$ then $\alpha \in F(S')$ for some finite subset $S' \subseteq S$.
5. An operator $F$ is called *progressive* if, for all subsets $S$ of $U$, $S \subseteq F(S)$.

**Examples**

- The immediate consequence operator $T_P$ for a definite logic program $P$ is an operator in $atoms(P)$ (the set of atoms of $P$), i.e. $T_P\colon \wp(atoms(P)) \to \wp(atoms(P))$.
The immediate consequence operator $T_P$ is
  - monotonic: $I_1 \subseteq I_2 \;\Rightarrow\; T_P(I_1) \subseteq T_P(I_2)$.
  - not progressive: e.g. $P = \{p \leftarrow q\}$. $T_P(\{q\}) = \{p\}$. $\{p\} \not\subseteq \{q\}$.
  - compact, because the body of every clause has a finite number of conditions. Spelled out in detail: if $A \in T_P(I)$ then there is a ground instance $A \leftarrow B_1, \dots, B_m$ of a clause in $P$ such that $\{B_1, \dots, B_m\} \subseteq I$. And so $A \in T_P(I')$ for a finite subset $I' = \{B_1, \dots, B_m\}$ of $I$.

We can also define an operator $T'_P(I) \overset{\text{def}}{=} I \cup T_P(I)$. $T'_P$ is obviously monotonic, progressive, and compact (if $P$ has a finite number of atoms).

- Classical propositional consequence $\mathrm{Th}\colon \wp(\mathcal{L}) \to \wp(\mathcal{L})$ maps sets of *formulas* of $\mathcal{L}$ to sets of *formulas* of $\mathcal{L}$.
Th is monotonic, progressive (since $W \subseteq \mathrm{Th}(W)$), and compact.

**Theorem (Knaster-Tarski lemma)** Let $F\colon \wp(U) \to \wp(U)$ be a monotonic operator.

1. The operator $F$ possesses a least fixpoint. This least fixpoint is equal to $F{\uparrow}^{\alpha}(\emptyset)$ for some ordinal $\alpha$.
2. If, in addition, $F$ is compact then the least fixpoint of $F$ is equal to $F{\uparrow}^{\omega}(\emptyset)$.

**Theorem** Let $F\colon \wp(U) \to \wp(U)$ be a monotonic and progressive operator. Then for every subset $X \subseteq U$, there is a least set $S$ such that $X \subseteq S$ and $S$ is a fixpoint of $F$. This set $S$ is of the form $F{\uparrow}^{\alpha}(X)$. If, in addition, $F$ is compact then this set $S$ is equal to $F{\uparrow}^{\omega}(X)$.

$$\begin{aligned} F{\uparrow}^{0}(X) &\overset{\text{def}}{=} X \\ F{\uparrow}^{n+1}(X) &\overset{\text{def}}{=} F(F{\uparrow}^{n}(X)) \\ F{\uparrow}^{\omega}(X) &\overset{\text{def}}{=} \textstyle\bigcup_{n\geq 0} F{\uparrow}^{n}(X) \end{aligned}$$

**Examples**

- Let $P$ be a definite logic program (i.e., no negation by failure not, no negation $\neg$).
$T_P(I) \subseteq I$ means that $I$ is a model of $P$. It is easy to check that $T_P(I) \subseteq I$ iff $T'_P(I) \subseteq I$.
Since $I \subseteq T'_P(I)$, the least (unique, smallest) model of $P$ is also the least fixpoint of $T'_P$, i.e. (Knaster-Tarski) $T'_P{\uparrow}^{\omega}(\emptyset)$.
The least (unique, smallest) model of $P$ that contains atoms $EDB$ is $T'_P{\uparrow}^{\omega}(EDB)$.

The original fixpoint semantics for definite logic program $P$ is given in terms of the least fixpoint of $T_P$, which is $T_P{\uparrow}^{\omega}$.
How does $T'_P{\uparrow}^{\omega}(\emptyset)$ compare to $T_P{\uparrow}^{\omega}$?
$T'_P{\uparrow}^{0}(\emptyset) = \emptyset = T_P{\uparrow}^{0}$. $T'_P{\uparrow}^{1}(\emptyset) = \emptyset \cup T_P(\emptyset) = T_P{\uparrow}^{1}$. $T'_P{\uparrow}^{2}(\emptyset) = T'_P(T'_P{\uparrow}^{1}(\emptyset)) = T_P(T'_P{\uparrow}^{1}(\emptyset)) \cup T'_P{\uparrow}^{1}(\emptyset) = T_P(T_P{\uparrow}^{1}) \cup T_P{\uparrow}^{1} = T_P{\uparrow}^{2} \cup T_P{\uparrow}^{1}$. And you can easily check (by induction on $n$) that

$$T'_P{\uparrow}^{n}(\emptyset) = \textstyle\bigcup_{i=0}^{n} T_P{\uparrow}^{n}$$

So then we have $T'_P{\uparrow}^{\omega}(\emptyset) \overset{\text{def}}{=} \bigcup_{n\geq 0} T'_P{\uparrow}^{n}(\emptyset) = \bigcup_{n\geq 0} T_P{\uparrow}^{n} \overset{\text{def}}{=} T_P{\uparrow}^{\omega}$.

The immediate consequence operator $T_P$ for definite logic programs has other properties (e.g. it is 'continuous') which we won't go into here.

- What is the least fixpoint of Th? Since Th is monotonic, progressive and compact, the least fixpoint is $\mathrm{Th}{\uparrow}^{\omega}(\emptyset)$. But since $\mathrm{Th}(\mathrm{Th}(S)) = \mathrm{Th}(S)$, $\mathrm{Th}{\uparrow}^{\omega}(\emptyset) = \mathrm{Th}(\emptyset)$. So the least fixpoint of Th is $\mathrm{Th}(\emptyset)$ (the set of all propositional tautologies). Exactly as we should expect. And the least fixpoint of Th containing the set of formulas $W$ is just $\mathrm{Th}(W)$. Again, exactly as we would expect.

**Definition** Let $F$ be an operator in $U$ and let $X$ be a subset of $U$. The closure of $X$ under the operator $F$ — denoted $Cl_F(X)$ — is the smallest subset $S$ of $U$ such that:

$$X \subseteq S \qquad \text{and} \qquad F(S) \subseteq S$$

How do we know that $Cl_F(X)$ exists and/or is unique?

**Proposition** $U$ satisfies the closure conditions (1)–(2) of $Cl_F(X)$.

**Proposition** If $F$ is monotonic, then if subsets $S_1$ and $S_2$ of $U$ both satisfy the closure conditions (1)–(2) of $Cl_F(W)$, then so does their intersection $S_1 \cap S_2$.

**Proposition** If $F$ is monotonic, $Cl_F(X)$ is the intersection of all sets $S$ satisfying the closure conditions (1)–(2).

$Cl_F(X)$ is the least set $S$ satisfying $X \subseteq S$ and $F(S) \subseteq S$. If in addition $F$ is progressive, $S \subseteq F(S)$. So then $Cl_F(X)$ is the least fixpoint of $F$ that contains $X$.

**Proposition** If $F$ is progressive, then $Cl_F(X)$ is the least fixpoint of $F$ that contains $X$. If $F$ is monotonic and progressive and compact, then $Cl_F(X) = F{\uparrow}^{\omega}(X)$.

(There is more about operators, closures, etc, later in the course.)

Note the notation and terminology:

- $Cl_{IDB}(EDB)$ — closure of *EDB* under the *rules IDB*
- $Cl_F(X)$ — closure of $X$ under the *operator* $F$

(They are different things. Sorry about that but it's convenient to use the same word and similar notation.)

**Example: definite deductive databases**

By definition, $Cl_{IDB}(EDB)$ is the least set of atoms $X$ such that

- $EDB \subseteq X$
- $T_{IDB}(X) \subseteq X$

It is very easy to check that $T_{IDB}(X) \subseteq X$ iff $T'_{IDB}(X) \subseteq X$.
(Because $A \subseteq B$ iff $A \cup B \subseteq B$.)

So, $Cl_{IDB}(EDB)$ is the least set of atoms $X$ such that

- $EDB \subseteq X$
- $T'_{IDB}(X) \subseteq X$

For definite clauses *IDB*, $T'_{IDB}$ is obviously progressive, monotonic and compact.
So then $Cl_{IDB}(EDB) = Cl_{T'_{IDB}}(EDB) = T'_{IDB} \uparrow^{\omega} (EDB)$.

Note the notation and terminology:

- $Cl_{IDB}(EDB)$ — closure of *EDB* under the *rules IDB*
- $Cl_{T'_{IDB}}(EDB)$ — closure of *EDB* under the *operator* $T'_{IDB}$

(I hope there's no confusion. Sorry about that but it's convenient.)

Note also

- $Cl_{IDB}(EDB) = Cl_{T_{IDB}}(EDB)$ (because $T_{IDB}(X) \subseteq X$ iff $T'_{IDB}(X) \subseteq X$)
- But $Cl_{T_{IDB}}(EDB) \neq T_{IDB} \uparrow^{\omega} (EDB)$ (because $T_{IDB}$ is not necessarily progressive).
  (However: $Cl_{IDB}(EDB) = Cl_{T_{IDB}}(EDB) = Cl_{T'_{IDB}}(EDB) = T'_{IDB} \uparrow^{\omega} (EDB)$.)

**Exercise** $Cl_{IDB}$ maps sets of atoms to sets of atoms. If *IDB* is *definite* then $Cl_{IDB}$ is a classical consequence operator (Tarski):

- $X \subseteq Cl_{IDB}(X)$
- $Cl_{IDB}(Cl_{IDB}(X)) \subseteq Cl_{IDB}(X)$
- if $X_1 \subseteq X_2$ then $Cl_{IDB}(X_1) \subseteq Cl_{IDB}(X_2)$