An interpretation is an assignment of values to all variables.

A model is an interpretation that satisfies the constraints.

Often we don’t want to just find a model, but want to know what is true in all models.

A proposition is statement that is true or false in each interpretation.
Why Propositions?

- Specifying logical formulae is often more natural than filling in tables.
- It is easier to check correctness and debug formulae than tables.
- We can exploit the Boolean nature for efficient reasoning.
- We need a language for asking queries (of what follows in all models) that may be more complicated than asking for the value of a variable.
- It is easy to incrementally add formulae.
- It can be extended to infinitely many variables with infinite domains (using logical quantification).
A Representation and Reasoning System (RRS) is made up of:

- **formal language**: specifies the legal sentences
- **semantics**: specifies the meaning of the symbols
- **reasoning theory or proof procedure**: nondeterministic specification of how an answer can be produced.
Using an RRS

1. Begin with a task domain.
2. Distinguish those things you want to talk about (the ontology).
3. Choose symbols in the computer to denote propositions
4. Tell the system knowledge about the domain.
5. Ask the system questions.
6. — the system can tell you whether your question is true
Role of semantics

In Computer:

\[ l_1\text{\_broken} \leftarrow \text{sw\_up} \]
\[ \land \text{power} \land \text{unlit\_l1}. \]
\[ \text{sw\_up}. \]
\[ \text{power} \leftarrow \text{lit\_l2}. \]
\[ \text{unlit\_l1}. \]
\[ \text{lit\_l2}. \]

In user's mind:

- \( l_1\text{\_broken} \): light \( l_1 \) is broken
- \( \text{sw\_up} \): switch is up
- \( \text{power} \): there is power in the building
- \( \text{unlit\_l1} \): light \( l_1 \) isn’t lit
- \( \text{lit\_l2} \): light \( l_2 \) is lit

Conclusion: \( l_1\text{\_broken} \)

- The computer doesn’t know the meaning of the symbols
- The user can interpret the symbol using their meaning
An **atom** is a symbol starting with a lower case letter.

A **body** is an atom or is of the form $b_1 \land b_2$ where $b_1$ and $b_2$ are bodies.

A **definite clause** is an atom or is a rule of the form $h \leftarrow b$ where $h$ is an atom and $b$ is a body.

A **knowledge base** is a set of definite clauses.
Semantics

- An interpretation $I$ assigns a truth value to each atom.
- A body $b_1 \land b_2$ is true in $I$ if $b_1$ is true in $I$ and $b_2$ is true in $I$.
- A rule $h \leftarrow b$ is false in $I$ if $b$ is true in $I$ and $h$ is false in $I$. The rule is true otherwise.
- A knowledge base $KB$ is true in $I$ if and only if every clause in $KB$ is true in $I$. 
A **model** of a set of clauses is an interpretation in which all the clauses are *true*.

If \( KB \) is a set of clauses and \( g \) is a conjunction of atoms, \( g \) is a **logical consequence** of \( KB \), written \( KB \models g \), if \( g \) is *true* in every model of \( KB \).

That is, \( KB \models g \) if there is no interpretation in which \( KB \) is *true* and \( g \) is *false*. 
Simple Example

\[ KB = \begin{cases} 
  p \leftarrow q. \\
  q. \\
  r \leftarrow s. 
\end{cases} \]

\[
\begin{array}{c|cccc}
  & p & q & r & s \\
  l_1 & true & true & true & true & \text{is a model of } KB \\
  l_2 & false & false & false & false & \text{not a model of } KB \\
  l_3 & true & true & false & false & \text{is a model of } KB \\
  l_4 & true & true & true & false & \text{is a model of } KB \\
  l_5 & true & true & false & true & \text{not a model of } KB \\
\end{array}
\]

\[ KB \models p, \ KB \models q, \ KB \not\models r, \ KB \not\models s \]
1. Choose a task domain: intended interpretation.
2. Associate an atom with each proposition you want to represent.
3. Tell the system clauses that are true in the intended interpretation: axiomatizing the domain.
4. Ask questions about the intended interpretation.
5. If $KB \models g$, then $g$ must be true in the intended interpretation.
6. The user can interpret the answer using their intended interpretation of the symbols.
The computer doesn’t have access to the intended interpretation.

All it knows is the knowledge base.

The computer can determine if a formula is a logical consequence of KB.

If $KB \models g$ then $g$ must be true in the intended interpretation.

If $KB \models g$ then there is a model of $KB$ in which $g$ is false. This could be the intended interpretation.
Representing the Electrical Environment

\[\begin{align*}
\text{light}_l_1. & \quad \text{lit}_l_1 \leftarrow \text{live}_w_0 \land \text{ok}_l_1 \\
\text{light}_l_2. & \quad \text{live}_w_0 \leftarrow \text{live}_w_1 \land \text{up}_s_2. \\
\text{down}_s_1. & \quad \text{live}_w_0 \leftarrow \text{live}_w_2 \land \text{down}_s_2. \\
\text{up}_s_2. & \quad \text{live}_w_1 \leftarrow \text{live}_w_3 \land \text{up}_s_1. \\
\text{up}_s_3. & \quad \text{live}_w_2 \leftarrow \text{live}_w_3 \land \text{down}_s_1. \\
\text{ok}_l_1. & \quad \text{lit}_l_2 \leftarrow \text{live}_w_4 \land \text{ok}_l_2. \\
\text{ok}_l_2. & \quad \text{live}_w_4 \leftarrow \text{live}_w_3 \land \text{up}_s_3. \\
\text{ok}_cb_1. & \quad \text{live}_p_1 \leftarrow \text{live}_w_3. \\
\text{ok}_cb_2. & \quad \text{live}_w_3 \leftarrow \text{live}_w_5 \land \text{ok}_cb_1. \\
\text{live}_\text{outside}. & \quad \text{live}_p_2 \leftarrow \text{live}_w_6. \\
& \quad \text{live}_w_6 \leftarrow \text{live}_w_5 \land \text{ok}_cb_2. \\
& \quad \text{live}_w_5 \leftarrow \text{live}_\text{outside}. 
\end{align*}\]
A **proof** is a mechanically derivable demonstration that a formula logically follows from a knowledge base.

Given a proof procedure, $KB \vdash g$ means $g$ can be derived from knowledge base $KB$.

Recall $KB \models g$ means $g$ is true in all models of $KB$.

A proof procedure is **sound** if $KB \vdash g$ implies $KB \models g$.

A proof procedure is **complete** if $KB \models g$ implies $KB \vdash g$. 
One rule of derivation, a generalized form of *modus ponens*:

If “$h \leftarrow b_1 \land \ldots \land b_m$” is a clause in the knowledge base, and each $b_i$ has been derived, then $h$ can be derived.

This is forward chaining on this clause.

(This rule also covers the case when $m = 0$.)
Bottom-up proof procedure

$KB \vdash g$ if $g \in C$ at the end of this procedure:

1. $C := \{\}$;
2. repeat
   1. select clause “$h \leftarrow b_1 \land \ldots \land b_m$” in $KB$ such that $b_i \in C$ for all $i$, and $h \notin C$;
   2. $C := C \cup \{h\}$
3. until no more clauses can be selected.
Example

\[ a \leftarrow b \land c. \]
\[ a \leftarrow e \land f. \]
\[ b \leftarrow f \land k. \]
\[ c \leftarrow e. \]
\[ d \leftarrow k. \]
\[ e. \]
\[ f \leftarrow j \land e. \]
\[ f \leftarrow c. \]
\[ j \leftarrow c. \]
Soundness of bottom-up proof procedure

If $KB \vdash g$ then $KB \models g$.

- Suppose there is a $g$ such that $KB \vdash g$ and $KB \not\models g$.
- Then there must be a first atom added to $C$ that isn’t true in every model of $KB$. Call it $h$. Suppose $h$ isn’t true in model $I$ of $KB$.
- There must be a clause in $KB$ of form

  $$ h \leftarrow b_1 \land \ldots \land b_m $$

  Each $b_i$ is true in $I$. $h$ is false in $I$. So this clause is false in $I$. Therefore $I$ isn’t a model of $KB$.
- Contradiction.
The $C$ generated at the end of the bottom-up algorithm is called a **fixed point**.

Let $I$ be the interpretation in which every element of the fixed point is true and every other atom is false.

$I$ is a model of $KB$.

Proof: suppose $h \leftarrow b_1 \land \ldots \land b_m$ in $KB$ is false in $I$. Then $h$ is false and each $b_i$ is true in $I$. Thus $h$ can be added to $C$. Contradiction to $C$ being the fixed point.

$I$ is called a **Minimal Model**.
If $KB \models g$ then $KB \vdash g$.

- Suppose $KB \models g$. Then $g$ is true in all models of $KB$.
- Thus $g$ is true in the minimal model.
- Thus $g$ is in the fixed point.
- Thus $g$ is generated by the bottom up algorithm.
- Thus $KB \vdash g$. 
Top-down Ground Proof Procedure

Idea: search backward from a query to determine if it is a logical consequence of $KB$.

An answer clause is of the form:

$$yes \leftarrow a_1 \land a_2 \land \ldots \land a_m$$

The SLD Resolution of this answer clause on atom $a_i$ with the clause:

$$a_i \leftarrow b_1 \land \ldots \land b_p$$

is the answer clause

$$yes \leftarrow a_1 \land \ldots \land a_{i-1} \land b_1 \land \ldots \land b_p \land a_{i+1} \land \ldots \land a_m.$$
Derivations

- An **answer** is an answer clause with $m = 0$. That is, it is the answer clause $yes ←$.

- A **derivation** of query "$q_1 \land \ldots \land q_k$" from $KB$ is a sequence of answer clauses $\gamma_0, \gamma_1, \ldots, \gamma_n$ such that
  - $\gamma_0$ is the answer clause $yes ← q_1 \land \ldots \land q_k$,
  - $\gamma_i$ is obtained by resolving $\gamma_{i-1}$ with a clause in $KB$, and
  - $\gamma_n$ is an answer.
To solve the query \(?q_1 \land \ldots \land q_k\):

\[ ac := \text{"yes} \leftarrow q_1 \land \ldots \land q_k\text{"} \]

repeat

\begin{align*}
\text{select} & \text{ atom } a_i \text{ from the body of } ac; \\
\text{choose} & \text{ clause } C \text{ from } KB \text{ with } a_i \text{ as head; } \\
& \text{replace } a_i \text{ in the body of } ac \text{ by the body of } C \\
\text{until} & \text{ ac is an answer.}
\end{align*}
Nondeterministic Choice

- **Don’t-care nondeterminism** If one selection doesn’t lead to a solution, there is no point trying other alternatives. 

- **Don’t-know nondeterminism** If one choice doesn’t lead to a solution, other choices may.
Example: successful derivation

\[ a \leftarrow b \land c. \quad a \leftarrow e \land f. \quad b \leftarrow f \land k. \]
\[ c \leftarrow e. \quad d \leftarrow k. \quad e. \]
\[ f \leftarrow j \land e. \quad f \leftarrow c. \quad j \leftarrow c. \]

Query: \(?a\)

\[ \gamma_0: \text{yes} \leftarrow a \quad \gamma_4: \text{yes} \leftarrow e \]
\[ \gamma_1: \text{yes} \leftarrow e \land f \quad \gamma_5: \text{yes} \leftarrow \]
\[ \gamma_2: \text{yes} \leftarrow f \]
\[ \gamma_3: \text{yes} \leftarrow c \]
Example: failing derivation

\[
\begin{align*}
a & \leftarrow b \land c. \\
c & \leftarrow e. \\
f & \leftarrow j \land e. \\
a & \leftarrow e \land f. \\
c & \leftarrow e. \\
\end{align*}
\]

\[
\begin{align*}
b & \leftarrow f \land k. \\
d & \leftarrow k. \\
f & \leftarrow c. \\
e. \\
j & \leftarrow c. \\
\end{align*}
\]

Query: \( ?a \)

\[
\begin{align*}
\gamma_0 : & \quad \text{yes} \leftarrow a \\
\gamma_1 : & \quad \text{yes} \leftarrow b \land c \\
\gamma_2 : & \quad \text{yes} \leftarrow f \land k \land c \\
\gamma_3 : & \quad \text{yes} \leftarrow c \land k \land c \\
\gamma_4 : & \quad \text{yes} \leftarrow e \land k \land c \\
\gamma_5 : & \quad \text{yes} \leftarrow k \land c \\
\end{align*}
\]
Search Graph for SLD Resolution

$$a \leftarrow b \land c. \quad a \leftarrow g.$$  
$$a \leftarrow h. \quad b \leftarrow j.$$  
$$b \leftarrow k. \quad d \leftarrow m.$$  
$$d \leftarrow p. \quad f \leftarrow m.$$  
$$f \leftarrow p. \quad g \leftarrow m.$$  
$$g \leftarrow f. \quad k \leftarrow m.$$  
$$h \leftarrow m. \quad p.$$  
$$?a \land d$$
Agents acting in an environment

Agent

Abilities
Goals/Preferences
Prior Knowledge
Observations
Past Experiences

Environment

Actions
In the electrical domain, what should the house builder know?

What should an occupant know?
In the electrical domain, what should the house builder know?

What should an occupant know?

Users can’t be expected to volunteer knowledge:
  ▶ They don’t know what information is needed.
  ▶ They don’t know what vocabulary to use.
Ask-the-user

- Users can provide observations to the system. They can answer specific queries.
- **Askable** atoms are those that a user should be able to observe.
- There are 3 sorts of goals in the top-down proof procedure:
  - Goals for which the user isn’t expected to know the answer.
  - Askable atoms that may be useful in the proof.
  - Askable atoms that the user has already provided information about.

The top-down proof procedure can be modified to ask users about askable atoms they have not already provided answers for.
Users can provide observations to the system. They can answer specific queries.

Askable atoms are those that a user should be able to observe.

There are 3 sorts of goals in the top-down proof procedure:

- Goals for which the user isn’t expected to know the answer.
- Askable atoms that may be useful in the proof.
- Askable atoms that the user has already provided information about.

The top-down proof procedure can be modified to ask users about askable atoms they have not already provided answers for.
Knowledge-Level Explanation

- **HOW** questions can be used to ask how an atom was proved. It gives the rule used to prove the atom. You can ask HOW an element of the body of that rules was proved. This lets the user explore the proof.

- **WHY** questions can be used to ask why a question was asked. It provides the rule with the asked atom in the body. You can ask WHY the rule in the head was asked.
Knowledge-Level Debugging

There are four types of non-syntactic errors that can arise in rule-based systems:

- An incorrect answer is produced: an atom that is false in the intended interpretation was derived.
- Some answer wasn’t produced: the proof failed when it should have succeeded. Some particular true atom wasn’t derived.
- The program gets into an infinite loop.
- The system asks irrelevant questions.
Debugging incorrect answers

- Suppose atom $g$ was proved but is false in the intended interpretation.
- There must be a rule $g \leftarrow a_1 \land \ldots \land a_k$ in the knowledge base that was used to prove $g$.
- Either:
  - one of the $a_i$ is false in the intended interpretation or
  - all of the $a_i$ are true in the intended interpretation.
- Incorrect answers can be debugged by only answering yes/no questions.
Electrical Environment

- **Light**
- **Two-way switch**
- **Switch**
- **Power outlet**
- **Circuit breaker**
- **Outside power**

Diagram:

- Line from outside power to `cb1`
- Line from `cb1` to `w5`
- Line from `w5` to `s1`
- Line from `s1` to `w1`
- Line from `w1` to `s2`
- Line from `s2` to `w2`
- Line from `w2` to `w3`
- Line from `w3` to `s3`
- Line from `s3` to `w4`
- Line from `w4` to `w6`
- Line from `w6` to `p1`
- Line from `p1` to `l1`
- Line from `l1` to `l2`
- Line from `l2` to `p2`
- Line from `p2` to `cb2`
- Line from `cb2` to `w6`
- Line from `w6` to `p2`
Missing Answers

If atom $g$ is true in the intended interpretation, but could not be proved, either:

- There is no appropriate rule for $g$.
- There is a rule $g \leftarrow a_1 \land \ldots \land a_k$ that should have succeeded.
Integrity Constraints

- In the electrical domain, what if we predict that a light should be on, but observe that it isn’t? What can we conclude?
- We will expand the definite clause language to include integrity constraints which are rules that imply false, where false is an atom that is false in all interpretations.
- This will allow us to make conclusions from a contradiction.
- A definite clause knowledge base is always consistent. This won’t be true with the rules that imply false.
Horn clauses

- An **integrity constraint** is a clause of the form
  $$false \leftarrow a_1 \land \ldots \land a_k$$
  where the $a_i$ are atoms and $false$ is a special atom that is false in all interpretations.

- A **Horn clause** is either a definite clause or an integrity constraint.
Negations can follow from a Horn clause KB.
The negation of \( \alpha \), written \( \neg \alpha \) is a formula that
- is true in interpretation \( I \) if \( \alpha \) is false in \( I \), and
- is false in interpretation \( I \) if \( \alpha \) is true in \( I \).

Example:

\[
KB = \begin{cases}
\text{false} \leftarrow a \land b. \\
a \leftarrow c. \\
b \leftarrow c.
\end{cases}
\]

\( KB \models \neg c. \)
Disjunctive Conclusions

- Disjunctions can follow from a Horn clause KB.
- The disjunction of $\alpha$ and $\beta$, written $\alpha \lor \beta$, is
  - true in interpretation $I$ if $\alpha$ is true in $I$ or $\beta$ is true in $I$
    (or both are true in $I$).
  - false in interpretation $I$ if $\alpha$ and $\beta$ are both false in $I$.

**Example:**

$$KB = \left\{ \begin{array}{l}
  \text{false} \leftarrow a \land b. \\
  a \leftarrow c. \\
  b \leftarrow d.
\end{array} \right\} \quad KB \models \neg c \lor \neg d.$$
Questions and Answers in Horn KBs

- An **assumable** is an atom whose negation you are prepared to accept as part of a (disjunctive) answer.

- A **conflict** of $KB$ is a set of assumables that, given $KB$ imply $false$.

- A **minimal conflict** is a conflict such that no strict subset is also a conflict.
Example: If \{c, d, e, f, g, h\} are the assumables

\[
KB = \begin{cases}
\text{false} \leftarrow a \land b. \\
a \leftarrow c. \\
b \leftarrow d. \\
b \leftarrow e.
\end{cases}
\]

- \{c, d\} is a conflict
- \{c, e\} is a conflict
- \{c, d, e, h\} is a conflict
Assume that the user is able to observe whether a light is lit or dark and whether a power outlet is dead or live.

A light can’t be both lit and dark. An outlet can’t be both live and dead:

\[
\text{false} \leftarrow \text{dark}_l \land \text{lit}_l.
\]

\[
\text{false} \leftarrow \text{dark}_l \land \text{lit}_l.
\]

\[
\text{false} \leftarrow \text{dead}_p \land \text{live}_p.
\]

Assume the individual components are working correctly:

\[
\text{assumable ok}_l.
\]

\[
\text{assumable ok}_s.
\]

\[
\ldots
\]

Suppose switches \(s_1, s_2, \) and \(s_3\) are all up:

\[
\text{up}_s. \quad \text{up}_s. \quad \text{up}_s.
\]
Representing the Electrical Environment

\[
\begin{align*}
\text{light}_1 & \leftarrow \text{live}_w_0 \land \text{ok}_l_1. \\
\text{live}_w_0 & \leftarrow \text{live}_w_1 \land \text{up}_s_2 \land \text{ok}_s_2. \\
\text{light}_2 & \leftarrow \text{live}_w_0 \land \text{down}_s_2 \land \text{ok}_s_2. \\
\text{live}_w_1 & \leftarrow \text{live}_w_3 \land \text{up}_s_1 \land \text{ok}_s_1. \\
\text{live}_w_2 & \leftarrow \text{live}_w_3 \land \text{down}_s_1 \land \text{ok}_s_1. \\
\text{lit}_l_2 & \leftarrow \text{live}_w_4 \land \text{ok}_l_2. \\
\text{live}_w_4 & \leftarrow \text{live}_w_3 \land \text{up}_s_3 \land \text{ok}_s_3. \\
\text{live}_p_1 & \leftarrow \text{live}_w_3. \\
\text{live}_w_3 & \leftarrow \text{live}_w_5 \land \text{ok}_c_b_1. \\
\text{live}_p_2 & \leftarrow \text{live}_w_6. \\
\text{live}_w_6 & \leftarrow \text{live}_w_5 \land \text{ok}_c_b_2. \\
\text{live}_w_5 & \leftarrow \text{live}_\text{outside}. 
\end{align*}
\]
If the user has observed $l_1$ and $l_2$ are both dark:

$$dark_1. \ dark_2.$$ 

There are two minimal conflicts:

$$\{ok\ cb_1, \ ok\ s_1, \ ok\ s_2, \ ok\ l_1\} \ \text{and} \ \{ok\ cb_1, \ ok\ s_3, \ ok\ l_2\}.$$ 

You can derive:

$$\neg ok\ cb_1 \ \vee \ \neg ok\ s_1 \ \vee \ \neg ok\ s_2 \ \vee \ \neg ok\ l_1$$

$$\neg ok\ cb_1 \ \vee \ \neg ok\ s_3 \ \vee \ \neg ok\ l_2.$$ 

Either $cb_1$ is broken or there is one of six double faults.
A **consistency-based diagnosis** is a set of assumables that has at least one element in each conflict.

A **minimal diagnosis** is a diagnosis such that no subset is also a diagnosis.

Intuitively, one of the minimal diagnoses must hold. A diagnosis holds if all of its elements are false.

**Example:** For the proceeding example there are seven minimal diagnoses: \{\textit{ok}_\textit{cb}_1\}, \{\textit{ok}_\textit{s}_1, \textit{ok}_\textit{s}_3\}, \{\textit{ok}_\textit{s}_1, \textit{ok}_\textit{l}_2\}, \{\textit{ok}_\textit{s}_2, \textit{ok}_\textit{s}_3\},...
Recall: top-down consequence finding

To solve the query $\forall q_1 \land \ldots \land q_k$:

$$ac := "yes \leftarrow q_1 \land \ldots \land q_k"$$

repeat

- select atom $a_i$ from the body of $ac$;
- choose clause $C$ from $KB$ with $a_i$ as head;
- replace $a_i$ in the body of $ac$ by the body of $C$

until $ac$ is an answer.
Implementing conflict finding: top down

- Query is \textit{false}.
- Don’t select an atom that is assumable.
- Stop when all of the atoms in the body of the generalised query are assumable:
  - this is a conflict
Example

\[
\begin{align*}
\text{false} & \leftarrow a. \\
a & \leftarrow b \& c. \\
b & \leftarrow d. \\
b & \leftarrow e. \\
c & \leftarrow f. \\
c & \leftarrow g. \\
e & \leftarrow h \& w. \\
e & \leftarrow g. \\
w & \leftarrow f. \\
\text{assumable } d, f, g, h.
\end{align*}
\]
**Conclusions** are pairs $\langle a, A \rangle$, where $a$ is an atom and $A$ is a set of assumables that imply $a$.

Initially, conclusion set $C = \{ \langle a, \{ a \} \rangle : a$ is assumable $\}$.

If there is a rule $h \leftarrow b_1 \land \ldots \land b_m$ such that for each $b_i$ there is some $A_i$ such that $\langle b_i, A_i \rangle \in C$, then $\langle h, A_1 \cup \ldots \cup A_m \rangle$ can be added to $C$.

If $\langle a, A_1 \rangle$ and $\langle a, A_2 \rangle$ are in $C$, where $A_1 \subset A_2$, then $\langle a, A_2 \rangle$ can be removed from $C$.

If $\langle false, A_1 \rangle$ and $\langle a, A_2 \rangle$ are in $C$, where $A_1 \subset A_2$, then $\langle a, A_2 \rangle$ can be removed from $C$. 
C := \{⟨a, \{a\}\} : a is assumable \};

repeat
    select clause "h ← b_1 ∧ \ldots ∧ b_m" in T such that
        ⟨b_i, A_i⟩ ∈ C for all i and
        there is no ⟨h, A'⟩ ∈ C or ⟨false, A'⟩ ∈ C
            such that A' ⊆ A where A = A_1 \cup \ldots \cup A_m;
    C := C ∪ \{⟨h, A⟩\}
    Remove any elements of C that can now be pruned;
until no more selections are possible
Often you want to assume that your knowledge is complete.

Example: you can state what switches are up and the agent can assume that the other switches are down.

Example: assume that a database of what students are enrolled in a course is complete.

The definite clause language is **monotonic:** adding clauses can’t invalidate a previous conclusion.

Under the complete knowledge assumption, the system is **non-monotonic:** adding clauses can invalidate a previous conclusion.
Suppose the rules for atom $a$ are

$$a \leftarrow b_1.$$  

...  

$$a \leftarrow b_n.$$  

equivalently $a \leftarrow b_1 \lor \ldots \lor b_n$.

Under the Complete Knowledge Assumption, if $a$ is true, one of the $b_i$ must be true:

$$a \rightarrow b_1 \lor \ldots \lor b_n.$$  

Under the CKA, the clauses for $a$ mean

$$a \leftrightarrow b_1 \lor \ldots \lor b_n.$$  

Clark’s completion:
Clark’s completion of a knowledge base consists of the completion of every atom.

If you have an atom \( a \) with no clauses, the completion is \( a \leftrightarrow \text{false} \).

You can interpret negations in the body of clauses. \( \sim a \) means that \( a \) is false under the complete knowledge assumption. This is **negation as failure**.
Bottom-up negation as failure interpreter

\[ C := \{\}; \]
repeat
  either
    select \( r \in KB \) such that
    \( r \) is “\( h \leftarrow b_1 \land \ldots \land b_m \)”
    \( b_i \in C \) for all \( i \), and
    \( h \notin C \);
    \( C := C \cup \{h\} \)
  or
    select \( h \) such that for every rule “\( h \leftarrow b_1 \land \ldots \land b_m \) \( \in KB \) 
    either for some \( b_i, \sim b_i \in C \)
    or some \( b_i = \sim g \) and \( g \in C \)
    \( C := C \cup \{\sim h\} \)
  until no more selections are possible
Negation as failure example

\[ p \leftarrow q \land \lnot r. \]
\[ p \leftarrow s. \]
\[ q \leftarrow \lnot s. \]
\[ r \leftarrow \lnot t. \]
\[ t. \]
\[ s \leftarrow w. \]
If the proof for $a$ fails, you can conclude $\sim a$.

Failure can be defined recursively:
Suppose you have rules for atom $a$:

$$a \leftarrow b_1$$
$$\vdots$$
$$a \leftarrow b_n$$

If each body $b_i$ fails, $a$ fails.
A body fails if one of the conjuncts in the body fails.
Note that you need finite failure. Example $p \leftarrow p$. 
Assumption-based Reasoning

Often we want our agents to make assumptions rather than doing deduction from their knowledge. For example:

- In **abduction** an agent makes assumptions to explain observations. For example, it hypothesizes what could be wrong with a system to produce the observed symptoms.

- In **default reasoning** an agent makes assumptions of normality to make predictions. For example, the delivery robot may want to assume Mary is in her office, even if it isn’t always true.
Design and Recognition

Two different tasks use assumption-based reasoning:

- **Design** The aim is to design an artifact or plan. The designer can select whichever design they like that satisfies the design criteria.

- **Recognition** The aim is to find out what is true based on observations. If there are a number of possibilities, the recognizer can’t select the one they like best. The underlying reality is fixed; the aim is to find out what it is.

**Compare:** Recognizing a disease with designing a treatment. Designing a meeting time with determining when it is.
The assumption-based framework is defined in terms of two sets of formulae:

- $F$ is a set of closed formula called the **facts**. These are formulae that are given as true in the world. We assume $F$ are Horn clauses.

- $H$ is a set of formulae called the **possible hypotheses** or **assumables**. Ground instance of the possible hypotheses can be assumed if consistent.
Making Assumptions

- A **scenario** of $\langle F, H \rangle$ is a set $D$ of ground instances of elements of $H$ such that $F \cup D$ is satisfiable.

- An **explanation** of $g$ from $\langle F, H \rangle$ is a scenario that, together with $F$, implies $g$.  
  
  $D$ is an explanation of $g$ if $F \cup D \models g$ and $F \cup D \nvDash false$.  
  
  A **minimal explanation** is an explanation such that no strict subset is also an explanation.

- An **extension** of $\langle F, H \rangle$ is the set of logical consequences of $F$ and a maximal scenario of $\langle F, H \rangle$.  

©D. Poole and A. Mackworth 2008  
Artificial Intelligence, Lecture 5.7, Page 4
Example

\[
a \leftarrow b \land c.
\]

\[
b \leftarrow e.
\]

\[
b \leftarrow h.
\]

\[
c \leftarrow g.
\]

\[
c \leftarrow f.
\]

\[
d \leftarrow g.
\]

\[
false \leftarrow e \land d.
\]

\[
f \leftarrow h \land m.
\]

\[
assymable \ e, h, g, m, n.
\]

\[
\{ e, m, n \} \text{ is a scenario.}
\]

\[
\{ e, g, m \} \text{ is not a scenario.}
\]

\[
\{ h, m \} \text{ is an explanation for } a.
\]

\[
\{ e, h, m \} \text{ is an explanation for } a.
\]

\[
\{ e, g, h, m \} \text{ isn’t an explanation.}
\]

\[
\{ e, g, h, m \} \text{ is a maximal scenario.}
\]

\[
\{ h, g, m, n \} \text{ is a maximal scenario.}
\]

\[
\{ h, g, m, n \} \text{ is a maximal scenario.}
\]

\[
\{ e, h, m, n \} \text{ is a maximal scenario.}
\]
Default Reasoning and Abduction

There are two strategies for using the assumption-based framework:

- **Default reasoning** Where the truth of \( g \) is unknown and is to be determined. An explanation for \( g \) corresponds to an argument for \( g \).

- **Abduction** Where \( g \) is given, and we are interested in explaining it. \( g \) could be an observation in a recognition task or a design goal in a design task.
Default Reasoning

- When giving information, you don’t want to enumerate all of the exceptions, even if you could think of them all.
- In default reasoning, you specify general knowledge and modularly add exceptions. The general knowledge is used for cases you don’t know are exceptional.
- Classical logic is **monotonic**: If $g$ logically follows from $A$, it also follows from any superset of $A$.
- Default reasoning is **nonmonotonic**: When you add that something is exceptional, you can’t conclude what you could before.
Defaults as Assumptions

Default reasoning can be modeled using

- $H$ is normality assumptions
- $F$ states what follows from the assumptions

An explanation of $g$ gives an argument for $g$. 
A reader of newsgroups may have a default: “Articles about AI are generally interesting”.

\[ H = \{ \text{int}_\text{ai}(X) \}, \]

where \( \text{int}_\text{ai}(X) \) means \( X \) is interesting if it is about AI. With facts:

\[
\text{interesting}(X) \leftarrow \text{about}_\text{ai}(X) \land \text{int}_\text{ai}(X).
\]

\( \text{about}_\text{ai}(\text{art}_23) \).

\( \{ \text{int}_\text{ai}(\text{art}_23) \} \) is an explanation for \( \text{interesting}(\text{art}_23) \).
We can have exceptions to defaults:

\[ \text{false} \leftarrow \text{interesting}(X) \land \text{uninteresting}(X). \]

Suppose article 53 is about AI but is uninteresting:

\[ \text{about}_{\text{ai}}(\text{art}_{\text{53}}). \]
\[ \text{uninteresting}(\text{art}_{\text{53}}). \]

We cannot explain \text{interesting}(\text{art}_{\text{53}}) even though everything we know about \text{art}_{\text{23}} you also know about \text{art}_{\text{53}}.
Exceptions to defaults

- interesting
- uninteresting
- int_ai
- about_ai
- class
- membership
- implication
- default

article_23
article_53
“Articles about formal logic are about Al.”
“Articles about formal logic are uninteresting.”
“Articles about machine learning are about Al.”

\[
\begin{align*}
  about_{-}ai(X) & \leftarrow about_{-}fl(X). \\
  uninteresting(X) & \leftarrow about_{-}fl(X). \\
  about_{-}ai(X) & \leftarrow about_{-}ml(X). \\
  about_{-}fl(art_{-}77). \\
  about_{-}ml(art_{-}34). \\
\end{align*}
\]

You can’t explain \( interesting(art_{-}77) \).
You can explain \( interesting(art_{-}34) \).
Exceptions to Defaults

- int_ai
- interesting
- article_23
- intro_question
- article_99
- article_34
- article_77
- about_fl
- about_ml
- about_ai
- implication
- default
- class
- membership

© D. Poole and A. Mackworth 2007
Artificial Intelligence, Lecture 9.4, Page 7
Formal logic is uninteresting by default
Suppose formal logic articles aren’t interesting by default:

\[ H = \{ \text{unint}_\text{fl}(X), \text{int}_\text{ai}(X) \} \]

The corresponding facts are:

\[
\begin{align*}
\text{interesting}(X) & \leftarrow \text{about}_\text{ai}(X) \land \text{int}_\text{ai}(X). \\
\text{about}_\text{ai}(X) & \leftarrow \text{about}_\text{fl}(X). \\
\text{uninteresting}(X) & \leftarrow \text{about}_\text{fl}(X) \land \text{unint}_\text{fl}(X). \\
\text{about}_\text{fl}(\text{art}_77). \\
\text{uninteresting}(\text{art}_77) & \text{ has explanation } \{ \text{unint}_\text{fl}(\text{art}_77) \}. \\
\text{interesting}(\text{art}_77) & \text{ has explanation } \{ \text{int}_\text{ai}(\text{art}_77) \}.
\end{align*}
\]
Because \textit{art\_77} is about formal logic, the argument “\textit{art\_77} is interesting because it is about AI” shouldn’t be applicable.

This is an instance of preference for more specific defaults.

Arguments that articles about formal logic are interesting because they are about AI can be defeated by adding:

\[
\text{false} \leftarrow \text{about}\_\text{fl}(X) \land \text{int}\_\text{ai}(X).
\]

This is known as a cancellation rule.

You can no longer explain \textit{interesting}(\textit{art\_77}).
Diagram of the Default Example

- `about_fl`
  - `unint_fl`
- `int_ai`
- `about_ai`
- `intro_question`
  - `article_23`
  - `article_77`
  - `article_34`
  - `article_99`

- `interesting`
  - implication
  - default
  - class membership

©D. Poole and A. Mackworth 2007
Artificial Intelligence, Lecture 9.4, Page 11
What if incompatible goals can be explained and there are no cancellation rules applicable? What should we predict?

For example: what if introductory questions are uninteresting, by default?

This is the **multiple extension problem**.

Recall: an **extension** of $\langle F, H \rangle$ is the set of logical consequences of $F$ and a maximal scenario of $\langle F, H \rangle$. 
Competing Arguments

- interesting_to_mary
- interesting_to_fred
- ai_im
- nar_im
- nar_if
- about_ai
- non_academic_recreation
- l_ai
- s_nar
- about_learning
- about_skiing
- induction_page
- learning_to_ski
- ski_Whistler_page
Skeptical Default Prediction

- We predict $g$ if $g$ is in all extensions of $\langle F, H \rangle$.
- Suppose $g$ isn’t in extension $E$. As far as we are concerned $E$ could be the correct view of the world. So we shouldn’t predict $g$.
- If $g$ is in all extensions, then no matter which extension turns out to be true, we still have $g$ true.
- Thus $g$ is predicted even if an adversary gets to select assumptions, as long as the adversary is forced to select something. You do not predict $g$ if the adversary can pick assumptions from which $g$ can’t be explained.
Recall: logical consequence is defined as truth in all models. We can define default prediction as truth in all minimal models.

Suppose $M_1$ and $M_2$ are models of the facts. $M_1 <_H M_2$ if the hypotheses violated by $M_1$ are a strict subset of the hypotheses violated by $M_2$. That is:

$$\{ h \in H' : h \text{ is false in } M_1 \} \subset \{ h \in H' : h \text{ is false in } M_2 \}$$

where $H'$ is the set of ground instances of elements of $H$. 
Minimal Models and Minimal Entailment

- \( M \) is a \textbf{minimal model} of \( F \) with respect to \( H \) if \( M \) is a model of \( F \) and there is no model \( M_1 \) of \( F \) such that \( M_1 \prec_H M \).
- \( g \) is \textbf{minimally entailed} from \( \langle F, H \rangle \) if \( g \) is true in all minimal models of \( F \) with respect to \( H \).
- **Theorem:** \( g \) is minimally entailed from \( \langle F, H \rangle \) if and only if \( g \) is in all extensions of \( \langle F, H \rangle \).
Evidential and Causal Reasoning

Much reasoning in AI can be seen as evidential reasoning, (observations to a theory) followed by causal reasoning (theory to predictions).

- **Diagnosis** Given symptoms, evidential reasoning leads to hypotheses about diseases or faults, these lead via causal reasoning to predictions that can be tested.

- **Robotics** Given perception, evidential reasoning can lead us to hypothesize what is in the world, that leads via causal reasoning to actions that can be executed.
To combine evidential and causal reasoning, you can either

- Axiomatize from causes to their effects and
  - use abduction for evidential reasoning
  - use default reasoning for causal reasoning
- Axiomatize both
  - effects $\rightarrow$ possible causes (for evidential reasoning)
  - causes $\rightarrow$ effects (for causal reasoning)

use a single reasoning mechanism, such as default reasoning.
Combining abduction and default reasoning

Representation:
- Axiomatize causally using rules.
- Have normality assumptions (defaults) for prediction
- Other assumptions to explain observations

Reasoning:
- Given an observation, use all assumptions to explain observation (find base causes)
- Use normality assumptions to predict from base causes explanations.
Why is the infobot trying another information source? (Arrows are implications or defaults. Sources are assumable.)
error_message ← data_absent ∧ da_em.
another_source_tried ← data_absent ∧ da_ast
another_source_tried ← data_inadequate ∧ di_ast.
data_absent ← file_removed ∧ fr_da.
data_absent ← link_down ∧ ld_da.
assumable file_removed.
assumable link_down.
assumable data_inadequate.
Example: fire alarm

tampering $ightarrow$ alarm

alarm $ightarrow$ leaving

leaving $ightarrow$ report

fire $ightarrow$ smoke
assumable tampering.

assumable fire.

alarm ← tampering ∧ tampering_caused_alarm.

alarm ← fire ∧ fire_caused_alarm.

default tampering_caused_alarm.

default fire_caused_alarm.

smoke ← fire ∧ fire_caused_smoke.

default fire_caused_smoke.

leaving ← alarm ∧ alarm_caused_leaving.

default alarm_caused_leaving.

report ← leaving ∧ leaving_caused_report.

default leaving_caused_report.
If we observe *report* there are two minimal explanations:

- one with *tampering*
- one with *fire*

If we observed just *smoke* there is one explanation (containing *fire*). This explanation makes no predictions about tampering.

If we had observed *report* ∧ *smoke*, there is one minimal explanation, (containing *fire*).

- The smoke explains away the tampering. There is no need to hypothesise *tampering* to explain report.