Complete Knowledge Assumption

- 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.

Completion of a knowledge base

- Suppose the rules for atom a are

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

  equivalently $a \leftarrow b_1 \vee \ldots \vee b_n$.
- Under the Complete Knowledge Assumption, if a is true, one of the b_i must be true:

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

- Under the CKA, the clauses for a mean Clark's completion:

$$a \leftrightarrow b_1 \vee \ldots \vee b_n$$

Clark's Completion of a KB

- 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 $a \leftrightarrow \mathit{false}$.
- 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

$$p \leftarrow q \wedge \sim r.$$
$$p \leftarrow s.$$
$$q \leftarrow \sim s.$$
$$r \leftarrow \sim t.$$
$$t.$$
$$s \leftarrow w.$$

Top-Down negation as failure proof procedure

- If the proof for a fails, you can conclude $\sim a$.
- Failure can be defined recursively:
  Suppose you have rules for atom a:

$$\begin{aligned} a &\leftarrow b_1 \\ &\vdots \\ a &\leftarrow b_n \end{aligned}$$

  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 $p \leftarrow p$.