To build an interpreter for a language, we need to distinguish

- **Base language** the language of the RRS being implemented.
- **Metalanguage** the language used to implement the system.

They could even be the same language!
Let’s use the definite clause language as the base language and the metalanguage.

- We need to represent the base-level constructs in the metalanguage.
- We represent base-level terms, atoms, and bodies as meta-level terms.
- We represent base-level clauses as meta-level facts.
- In the non-ground representation base-level variables are represented as meta-level variables.
Representing the base level constructs

- Base-level atom \( p(t_1, \ldots, t_n) \) is represented as the meta-level term \( p(t_1, \ldots, t_n) \).
- Meta-level term \( oand(e_1, e_2) \) denotes the conjunction of base-level bodies \( e_1 \) and \( e_2 \).
- Meta-level constant \( true \) denotes the object-level empty body.
- The meta-level atom \( clause(h, b) \) is true if “\( h \) if \( b \)” is a clause in the base-level knowledge base.
The base-level clauses

\[
\begin{align*}
\text{connected\_to}(l_1, w_0). \\
\text{connected\_to}(w_0, w_1) &\leftarrow up(s_2). \\
lit(L) &\leftarrow light(L) \land ok(L) \land live(L).
\end{align*}
\]

\[\text{can be represented as the meta-level facts}\]

\[
\begin{align*}
\text{clause}(\text{connected\_to}(l_1, w_0), \text{true}). \\
\text{clause}(\text{connected\_to}(w_0, w_1), up(s_2)). \\
\text{clause}(\text{lit}(L), \text{oand}(\text{light}(L), \text{oand}(\text{ok}(L), \text{live}(L)))).
\end{align*}
\]
Making the representation pretty

- Use the infix function symbol “&” rather than oand.
  - instead of writing oand(e₁, e₂), you write e₁ & e₂.
- Instead of writing clause(h, b) you can write h ⇐ b, where ⇐ is an infix meta-level predicate symbol.
  - Thus the base-level clause “h ⇐ a₁ ∧ ··· ∧ aₙ” is represented as the meta-level atom h ⇐ a₁ & ··· & aₙ.
The base-level clauses

\[
\begin{align*}
\text{connected} \_\text{to}(l_1, w_0). \\
\text{connected} \_\text{to}(w_0, w_1) & \leftarrow \text{up}(s_2). \\
\text{lit}(L) & \leftarrow \text{light}(L) \land \text{ok}(L) \land \text{live}(L).
\end{align*}
\]

can be represented as the meta-level facts

\[
\begin{align*}
\text{connected} \_\text{to}(l_1, w_0) & \leftarrow \text{true}. \\
\text{connected} \_\text{to}(w_0, w_1) & \leftarrow \text{up}(s_2). \\
\text{lit}(L) & \leftarrow \text{light}(L) \land \text{ok}(L) \land \text{live}(L).
\end{align*}
\]
\( \text{prove}(G) \) is true when base-level body \( G \) is a logical consequence of the base-level KB.

\[
\text{prove}(\text{true}).
\]
\[
\text{prove}((A \& B)) \leftarrow \\
\text{prove}(A) \land \\
\text{prove}(B).
\]
\[
\text{prove}(H) \leftarrow \\
(H \iff B) \land \\
\text{prove}(B).
\]
Example base-level KB

\[
\begin{align*}
live(W) & \iff \\
& \quad connected\_to(W, W_1) \& \\
& \quad live(W_1). \\
live(outside) & \iff true. \\
connected\_to(w_6, w_5) & \iff ok(cb_2). \\
connected\_to(w_5, outside) & \iff true. \\
ok(cb_2) & \iff true. \\
?prove(live(w_6)).
\end{align*}
\]
Expanding the base-level

Adding clauses increases what can be proved.

- **Disjunction** Let $a; b$ be the base-level representation for the disjunction of $a$ and $b$. Body $a; b$ is true when $a$ is true, or $b$ is true, or both $a$ and $b$ are true.

- **Built-in predicates** You can add built-in predicates such as $N$ is $E$ that is true if expression $E$ evaluates to number $N$. 
\begin{itemize}
\item \texttt{prove(true)}.
\item \texttt{prove((A & B)) \leftarrow prove(A) \land prove(B)}.
\item \texttt{prove((A; B)) \leftarrow prove(A)}.
\item \texttt{prove((A; B)) \leftarrow prove(B)}.
\item \texttt{prove((N is E)) \leftarrow N is E}.
\item \texttt{prove(H) \leftarrow (H \iff B) \land prove(B)}.
\end{itemize}
Adding conditions reduces what can be proved.

\% $bprove(G, D)$ is true if $G$ can be proved with a proof tree of depth less than or equal to number $D$.

\[
\begin{align*}
    bprove(true, D).
    bprove((A \& B), D) & \leftarrow \\
    & \quad bprove(A, D) \land bprove(B, D).
    bprove(H, D) & \leftarrow \\
    & \quad D \geq 0 \land D_1 \text{ is } D - 1 \land \\
    & \quad (H \leftarrow B) \land bprove(B, D_1).
\end{align*}
\]