Formale Methoden der Softwaretechnik
Formal methods of software engineering

Till Mossakowski, Christoph Lüth

SoSe 2011

TW consequence $\neq$ FO consequence

We have encountered arguments that are valid in Tarski's World but not FO valid.

$$\begin{array}{|l} \forall x(\mathrm{Cube}(x) \leftrightarrow \mathrm{SameShape}(x, c)) \\ \hline \mathrm{Cube}(c) \end{array}$$

The replacement method yields an invalid argument:

$$\begin{array}{|l} \forall x(P(x) \leftrightarrow Q(x, c)) \\ \hline P(c) \end{array}$$

The axiomatic method

Axiomatic method: bridge the gap between Tarski's World validity and FO validity by systematically expressing facts about the meanings of the predicates, and introduce them as *axioms*. Axioms restrict the possible interpretation of predicates.
Axioms may be used as premises within arguments/proofs.

The argument revisited

$$
\begin{array}{|l}
\forall x(\mathrm{Cube}(x) \leftrightarrow \mathrm{SameShape}(x, c)) \\
\forall x \mathrm{SameShape}(x, x) \\
\hline
\mathrm{Cube}(c)
\end{array}
$$

The replacement method yields a valid argument:

$$
\begin{array}{|l}
\forall x(P(x) \leftrightarrow Q(x, c)) \\
\forall x Q(x, x) \\
\hline
P(c)
\end{array}
$$

Common Algebraic Specification Language

- strongly typed; types are declated using the *sort* keyword
  sort Blocks
- predicates have to be declared with their types
  preds Cube, Dodec, Tet : Blocks
  preds LeftOf, RightOf, SameShape : Blocks * Blocks
- propositional variables = nullary predicates
  preds A,B,C : ()
- constants have to be declared with their types
  ops a,b,c : Blocks

Example CASL specification: blocks

```
spec Tarski1 =     sort Blocks
  preds Cube, Dodec, Tet, Small, Medium, Large : Blocks
  ops a,b,c : Blocks
  . not a=b . not a=c  . not b=c
  . Small(a) => Cube(a)     %(small_cube_a)%
  . Small(a) <=> Small(b)   %(small_a_b)%
  . Small(b) \/ Medium(b)   %(small_medium_b)%
  . Medium(b) => Medium(c)  %(medium_b_c)%
  . Medium(c) => Tet(c)     %(medium_tet_c)%
  . not Tet(c)              %(not_tet_c)%
  . Cube(a)                 %(cube_a)% %implied
  . Cube(b)                 %(cube_b)% %implied
```