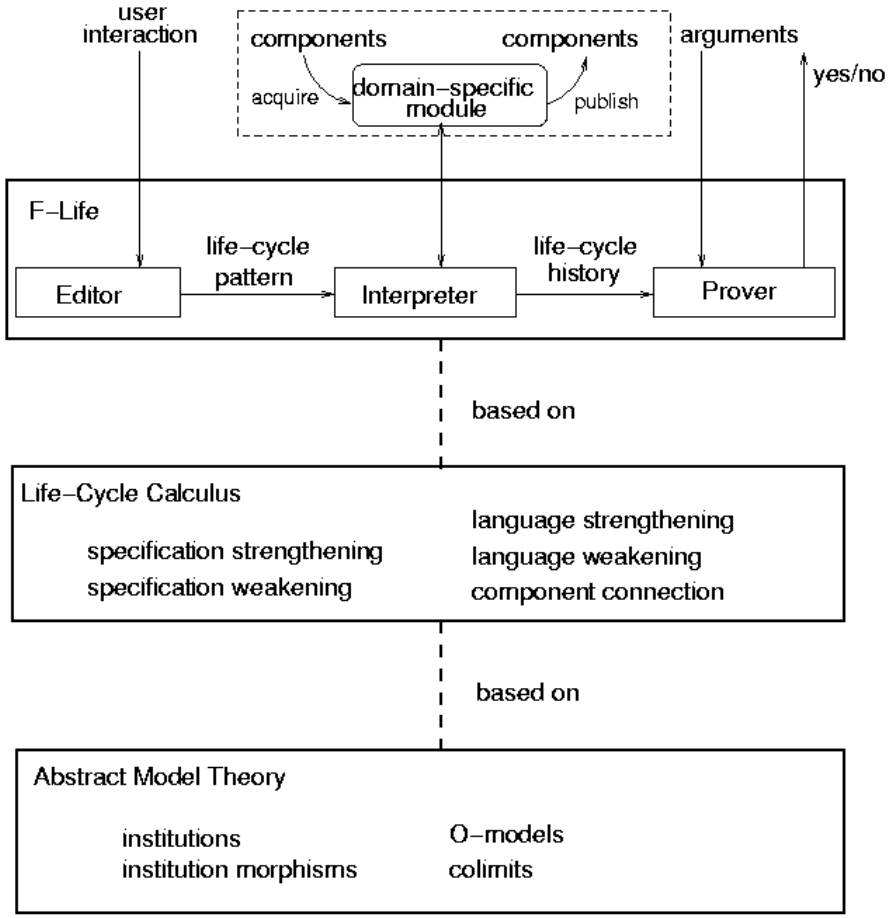
Formal Life-Cycle Specification with Generic Life-Cycle Ontologies

Marco Schorlemmer

17 June 2003

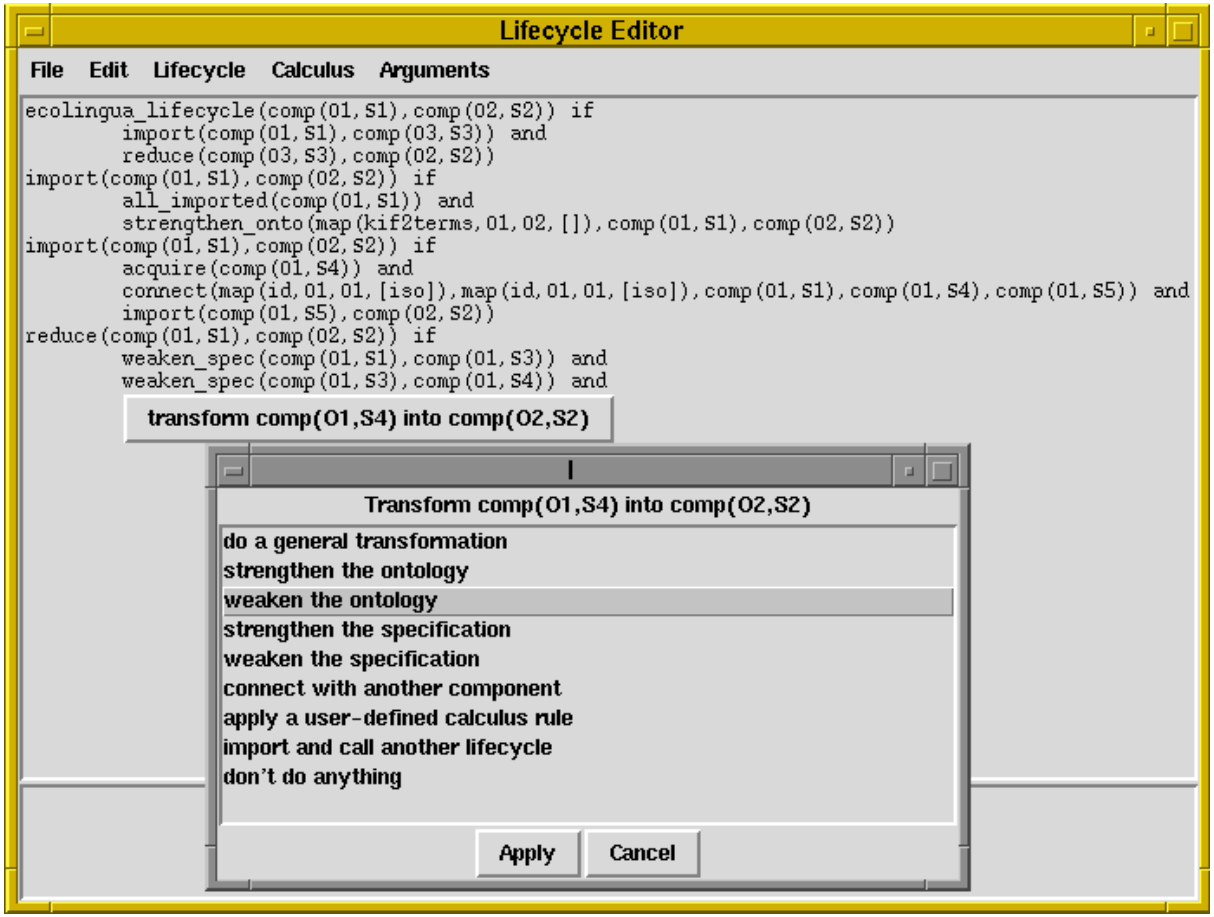
The Old F-Life Architecture



A Life-Cycle Calculus

Specification Strengthening (SS)	$\dfrac{\langle L, S\rangle}{\langle L, S'\rangle}$	if $S \sqsubset S'$
Specification Weakening (SW)	$\dfrac{\langle L, S\rangle}{\langle L, S'\rangle}$	if $S \sqsupset S'$
Language Strengthening (LS)	$\dfrac{\langle L, S\rangle}{\langle L', f[S]\rangle}$	if $L \xrightarrow{f} L'$
Language Weakening (LW)	$\dfrac{\langle L, S\rangle}{\langle L', f^{-1}[S]\rangle}$	if $L \xleftarrow{f} L'$
Component Connection (CC)	$\dfrac{\langle L_1, S_1\rangle \cdots \langle L_n, S_n\rangle}{\langle L, g_1[S_1] \sqcup \cdots \sqcup g_2[S_2]\rangle}$	if $g_{1\ldots n} : \{L_i \xrightarrow{f_{ij}} L_j\} \rhd L$

Editing Life-Cycle Patterns



Editing Life-Cycle Patterns

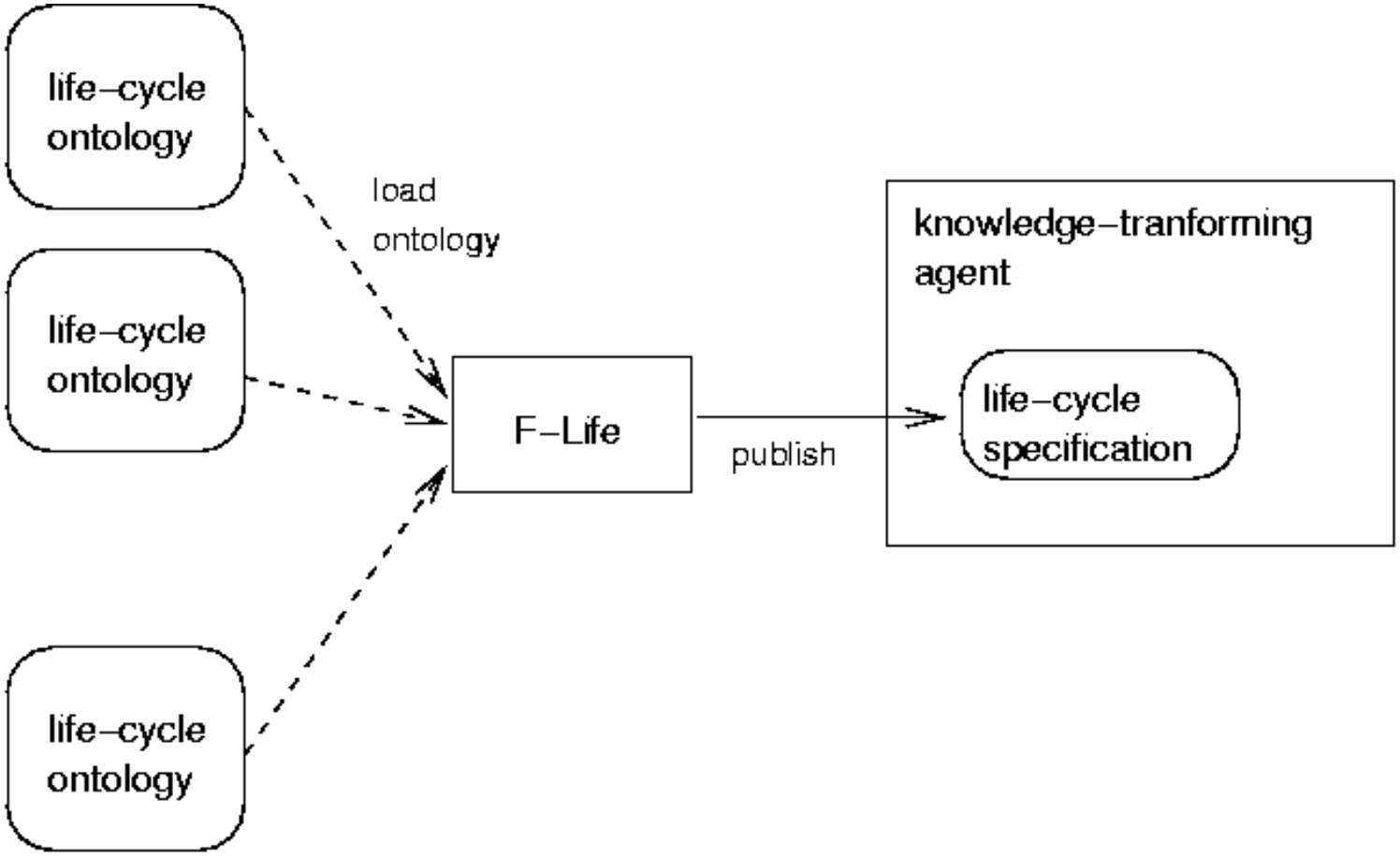
$$
\begin{aligned}
&ecolingua\_life\_cycle(\langle L_1, S_1\rangle, \langle L_2, S_2\rangle) \longleftarrow \\
&\qquad import(\langle L_1, S_1\rangle, \langle L_3, S_3\rangle) \wedge \\
&\qquad reduce(\langle L_3, S_3\rangle, \langle L_2, S_2\rangle)
\end{aligned}
$$

$$
\begin{aligned}
&import(\langle L_1, S_1\rangle, \langle L_2, S_2\rangle) \longleftarrow \\
&\qquad all\_imported(\langle L_1, S_1\rangle) \wedge \\
&\qquad strengthen\_lang(L_1 \overset{ol2terms}{\rightarrow} L_2, \langle L_1, S_1\rangle, \langle L_2, S_2\rangle)
\end{aligned}
$$

$$
\begin{aligned}
&import(\langle L_1, S_1\rangle, \langle L_2, S_2\rangle) \longleftarrow \\
&\qquad acquire(\langle L_1, S_4\rangle) \wedge \\
&\qquad connect(L_1 \overset{id}{\rightarrow} L_1, L_1 \overset{id}{\rightarrow} L_1, \langle L_1, S_1\rangle, \langle L_1, S_4\rangle, \langle L_1, S_5\rangle) \wedge \\
&\qquad import(\langle L_1, S_5\rangle, \langle L_2, S_2\rangle)
\end{aligned}
$$

$$
\begin{aligned}
&reduce(\langle L_1, S_1\rangle, \langle L_2, S_2\rangle) \longleftarrow \\
&\qquad weaken\_spec(\langle L_1, S_1\rangle, \langle L_1, S_3\rangle) \wedge \\
&\qquad weaken\_spec(\langle L_1, S_3\rangle, \langle L_1, S_4\rangle) \wedge \\
&\qquad weaken\_lang(L_1 \overset{terms2pl}{\leftarrow} L_2, \langle L_1, S_4\rangle, \langle L_2, S_2\rangle)
\end{aligned}
$$

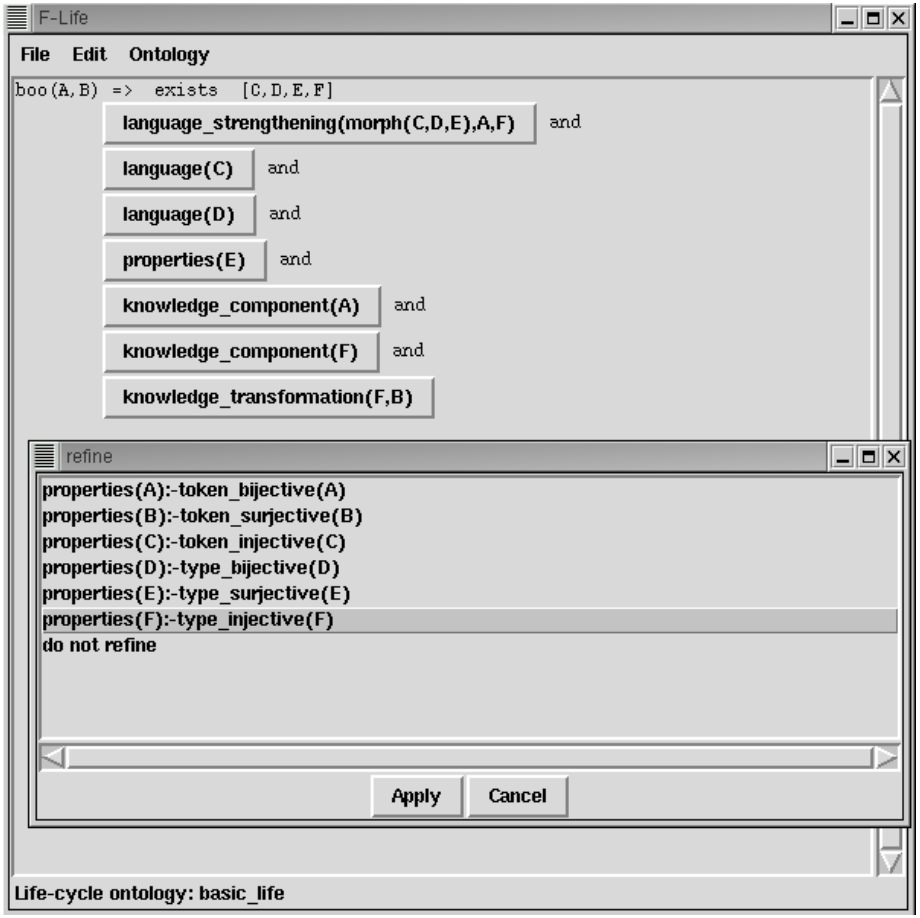
A New Architecture for F-Life



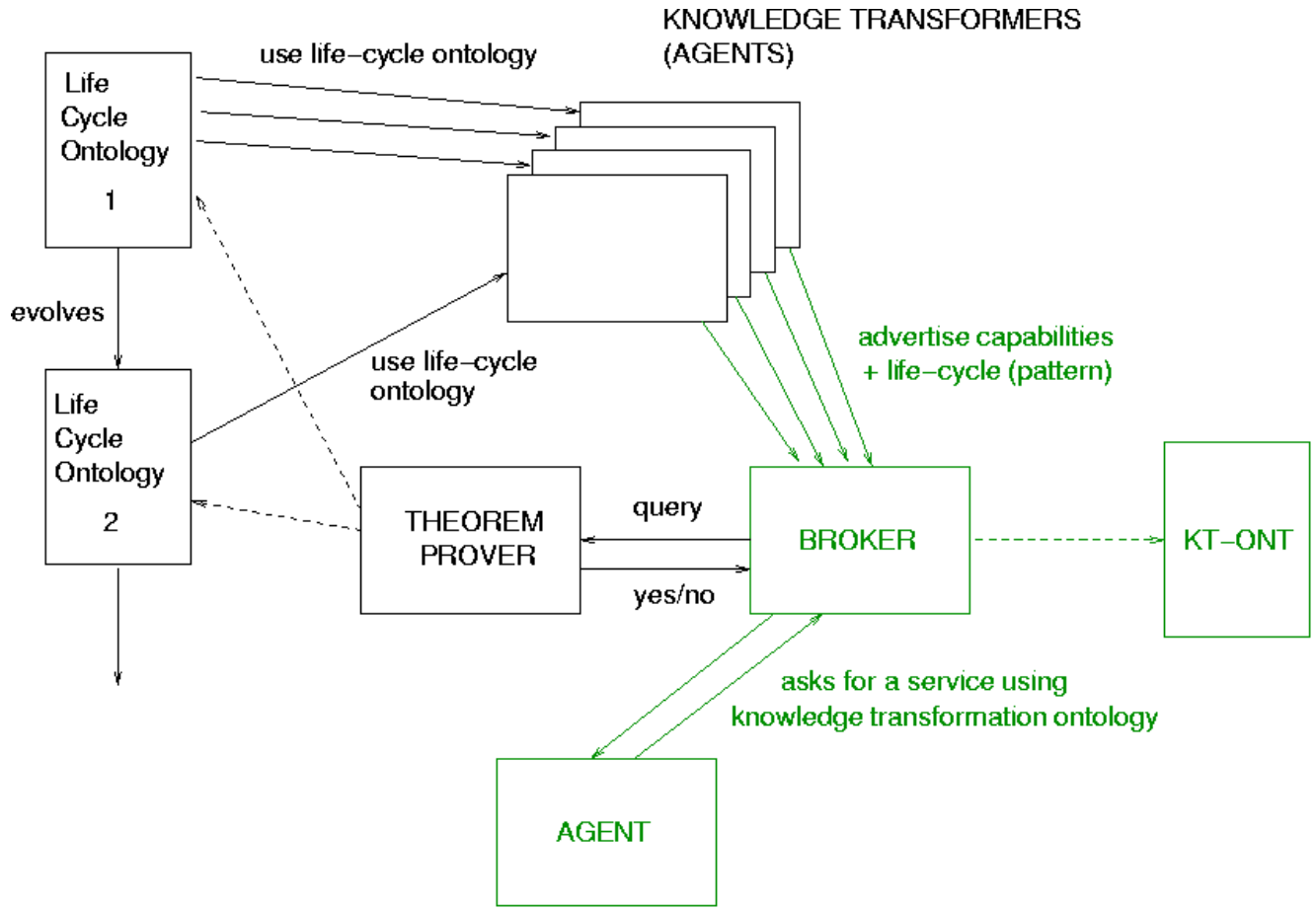
The Basic Life-Cycle Ontology (fragment)

```
knowledge_transformation(C1,C2) :-
        language_strengthening(F,C1,C2),
        language_morphism(F),
        knowledge_component(C1),
        knowledge_component(C2).

knowledge_transformation(C1,C2) :-
        language_weakening(F,C1,C2),
        language_morphism(F),
        knowledge_component(C1),
        knowledge_component(C2).

knowledge_transformation(C1,C2) :-
        knowledge_transformation(C1,C3),
        knowledge_transformation(C3,C2).
```

Life-Cycle Specification and Publishing in F-Life



F-Life and the Brokering of Knowledge Transformation Services



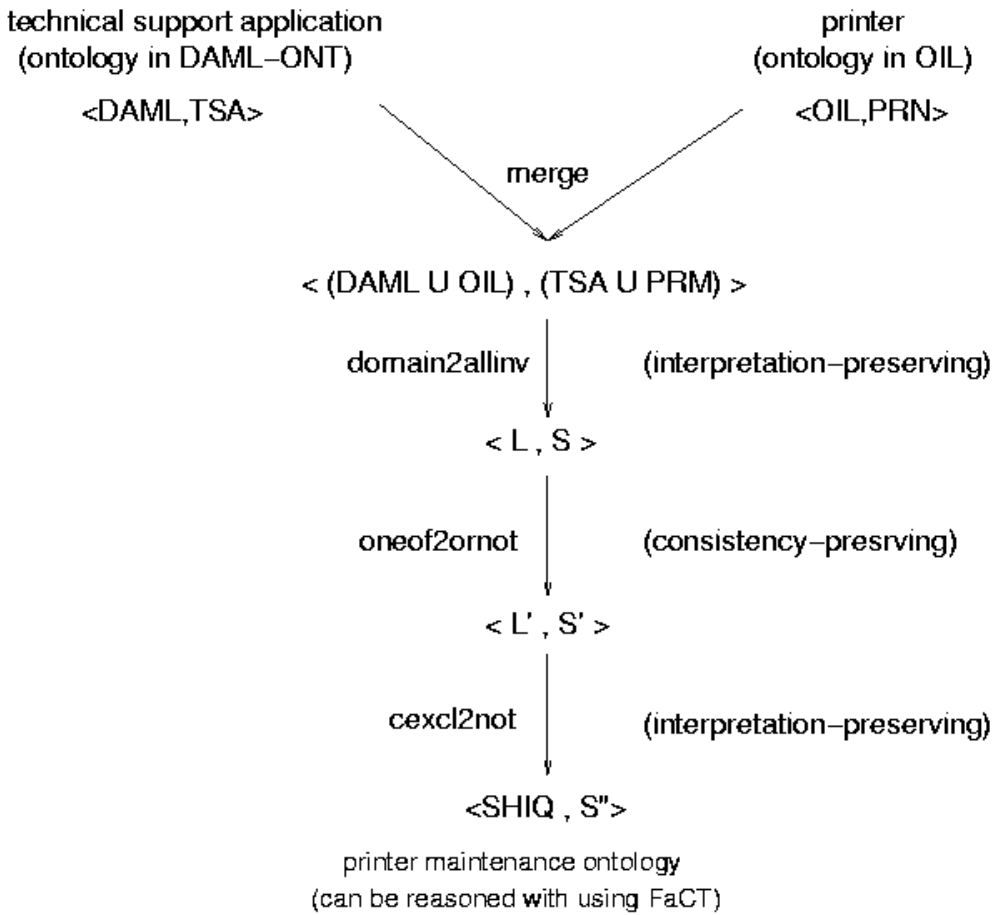
An Example: Language Translations

Language Translations as put forth in Euzenat & Stuckensmidt's ECAI-KTSW'02 paper

Knowledge transformations due to language translation:

- language iunclusion
- interpretation presering transformation
- expressivity-preserving transformation
- epimorphic transformation
- consequence-preserving transformation
- consistency-preserving transformation

A Realistic(?) Example



Euzenat & Stuckenschmidt's Transformations (fragment)

```
language_inclusion(C1,C2) :-
        language_strengthening(morph(_,_,P),C1,C2),
        type_inclusive(P).
language_inclusion(C1,C3) :-
        language_inclusion(C1,C2),
        language_inclusion(C2,C3).

interpretation_preserving_transformation(C1,C2) :-
        language_strengthening(morph(_,_,P),C1,C2),
        type_injective(P).
interpretation_preserving_transformation(C1,C3) :-
        interpretation_preserving_transformation(C1,C2),
        interpretation_preserving_transformation(C2,C3).
```

Knowledge-Transforming Agent 'boo'

```
lifecycle('basic_life.pl',
          boo(A,B) => [language_strengthening(morph(s1,t1,prop1),A,c),
                       language(s1),
                       language(t1),
                       type_inclusive(prop1),
                       knowledge_component(A),
                       knowledge_component(c),
                      language_strengthening(morph(s2,t2,prop2),c,B),
                       language(s2),
                       language(t2),
                       type_inclusive(prop2),
                       knowledge_component(c),
                       knowledge_component(B)]).
```

Querying the Prover for a Property

```
prove('euzenat.pl', 'boo.pl',
      boo(A,B) => [interpretation_preserving_transformation(A,B)])
```

`euzenat.pl` is the file containing the ontological definitions
`boo.pl` is the file containing the life-cycle specification

Future Work

- integration with the borker
- translators between ontology representations languages
- migration to Java
- experiment with more life-cycle ontologies and example scenarios