Description Logics

Description logics are formal knowledge representation languages with a relatively simple syntax and well-defined semantics. According to the description logic paradigm, knowledge is divided into a terminological part (TBox), where concepts like *"beverages that are carbonated and have some ingredient that is alcoholic"* are defined, and an assertional part (ABox), where individuals are related to each other and asserted as being instances of certain concepts.

A (partial) definition for the concept $Cider$, involving other concepts and the role (relation) $has\text{-}ingredient$ could be given as follows:

$$Cider \ \dot{\sqsubseteq} \ Beverage \sqcap Carbonated \sqcap \exists has\text{-}ingredient.Alcoholic$$

A corresponding ABox might contain the following assertions:

$$my\text{-}drink : Cider \qquad\qquad (my\text{-}drink, x) : has\text{-}ingredient$$

Reasoning in description logics:

- Is a given concept formula *consistent*?
- Does a concept *subsume* another one, i.e. is it more general than the other?
- Is a given ABox individual an *instance* of a given concept formula?
- Is a given ABox *consistent*?

Tableau-like calculi for description logics:

It turns out that all the above questions can be reduced to the question whether a given ABox is consistent. Tableau based calculi for description logics are similar to Tableaux for modal logics, but with named worlds, the ABox individuals.

Recently, research into optimisation techniques for description logics has been very active. Optimisations include:

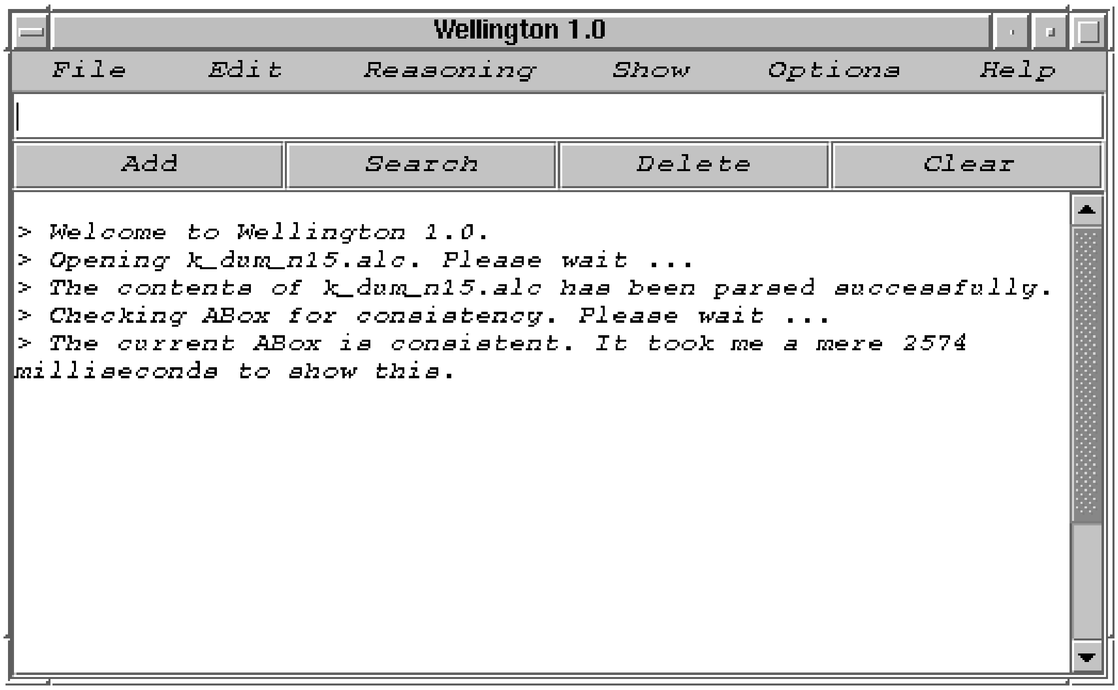
- *Semantic branching* with *branching heuristics* (e.g. prefer subformulas appearing often in short disjunctions);
- *Boolean constraint propagation:* avoid branching whenever possible;
- *Backjumping:* keep track of origin of formulas to be able to prune redundant branches; ...

Wellington 1.0

WELLINGTON 1.0 is the first public release of the description logics based knowledge representation and reasoning system developed by the Group of Logic and Computation at King's College London. To run or download the software please visit the project web site:

`http://www.dcs.kcl.ac.uk/research/groups/logic/wellington/`

Currently, WELLINGTON can be used to check the consistency of an ABox specified in the description logic $\mathcal{ALC}$, to check a concept formula for consistency, or to check whether one concept formula subsumes another one.



Main design objectives:

- *Extensibility* to different logics, calculi and their algorithmic variations by using open object-oriented data-structures;
- *Usability* for end-users through easy-to-use GUIs and for application programmers by providing appropriate Java libraries;
- *Efficiency* through use and further development of optimisation techniques.

* *The ABox stored in file k_dum_n15.alc (see screen shot) starts like this:*
(INSTANCE A (NOT (OR (OR (SOME R1 (SOME R1 (SOME R1 (SOME R1 (SOME R1 (SOME R1 (SOME R1 (SOME R1 (SOME R1 (SOME R1 (SOME R1 (SOME R1 (SOME R1 (SOME R1 (SOME R1 (SOME R1 (NOT (AND (AND (AND (OR (ALL R1 (ALL R1 (OR P0 (NOT (ALL R1 (OR (ALL R1 P0) (NOT P0))))))) (NOT (ALL R1 (OR P0 (NOT (ALL R1 (OR (ALL R1 P0) (NOT P0)))))))) (ALL R1 (OR (ALL R1 (ALL R1 P0)) (NOT (ALL R1 P0))))) (OR (OR (OR P0 (ALL R1 P0)) (NOT (SOME R1 (ALL R1 P0)))) (NOT (ALL R1 (OR P0 (NOT (ALL R1 (OR (ALL R1 P0) (NOT P0))))))))) (OR (OR (OR (OR (ALL R1 P0) (NOT P0)) (ALL R1 (OR (ALL R1 P0) (NOT P0)))) (NOT (SOME R1 (ALL R1 (OR (ALL R1 P0) (NOT P0)))))) (NOT (ALL R1 (OR (OR (ALL R1 P0) (NOT P0)) (NOT (ALL R1 (OR (ALL R1 (OR (ALL R1 P0) (NOT P0))) (NOT (OR (ALL R1 P0) (NOT P0))))))))))))))))))))))))))))))) (OR (OR (OR (OR (OR (SOME R1 (SOME R1 (SOME R1 (SOME R1 (SOME R1 (SOME R1 (SOME R1 (SOME R1 (SOME R1 (NOT (AND (ALL R1 (OR (ALL R1 (ALL R1 P0)) (NOT (ALL R1 P0)))) (OR (OR (OR P0 (ALL R1 P0)) (NOT (SOME R1 (ALL R1 P0)))) (NOT (ALL R1 (OR P0 (NOT (ALL ...

Current and Future Developments

WELLINGTON 1.0 is only the beginning. In the long run we intend to build a system offering a choice of algorithms for several languages. In particular, we want to integrate the following features:

- *TBox unfolding* and *general TBox axioms*.
- *Concrete domain reasoning.* In cooperation with the LIIA Strasbourg we have defined a general Java interface for concrete domains to be able to exchange modules. Amongst other things, we want to use this for reasoning about temporal intervals. Example:

$$EatingOrderObedient \ \dot{\sqsubseteq} \ \ before(starter \circ time, desert \circ time)^*$$

- *Arithmetical reasoning.* Ohlbach (1999) introduces a description logic with arithmetical constraints over sets of role-fillers. Example:

$$ShoeFetishist \doteq Person \ \sqcap \ |has\text{-}shoe| > 0.8 \times |has\text{-}sock|$$

  For this logic we intend to implement a theory resolution style calculus with mathematical programming as the background theory.
- *Classification* (determining the partial order over named concepts wrt. the subsumption relation).
- *Knowledge acquisition* (see poster by Stefan Schlobach).

The Data Driven Logic Algorithms Project

Data Driven Logic Algorithms is an EPSRC funded three year project based at King's College, which is focusing on the integration of description logics with various other fields, like mathematical programming, databases, artificial learning, and temporal reasoning. The people involved in the project are Hans Jürgen Ohlbach, Dov Gabbay, Odinaldo Rodrigues, Stefan Schlobach, and Ulrich Endriss.

One of the main objectives of the project is to develop a powerful hybrid system for reasoning with expressive description logics. WELLINGTON 1.0 will provide the basis for this venture.

*In the example, the roles $starter$ and $desert$ are meant to relate someone to the event of that person having his or her starter and desert, respectively.