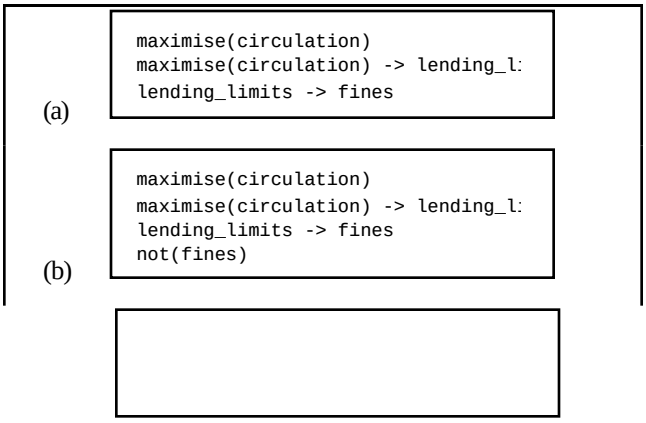
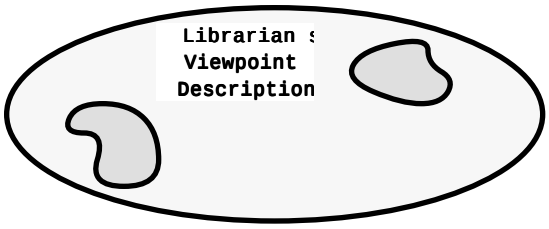
concerns of requirements engineering and knowledge acquisition. Current research in the latter focuses on models of problem solving behaviour, with the goal of implementing systems that demonstrate such behaviour. Emphasis is placed on imitation of an expert's performance. No attempt is made to model the social and organisational setting in which the behaviour is embedded, as is consistent with a purely cognitive stance. On the other hand, much of software engineering deals with systems that support human activities, and hence an understanding of the social and organisation setting is crucial.

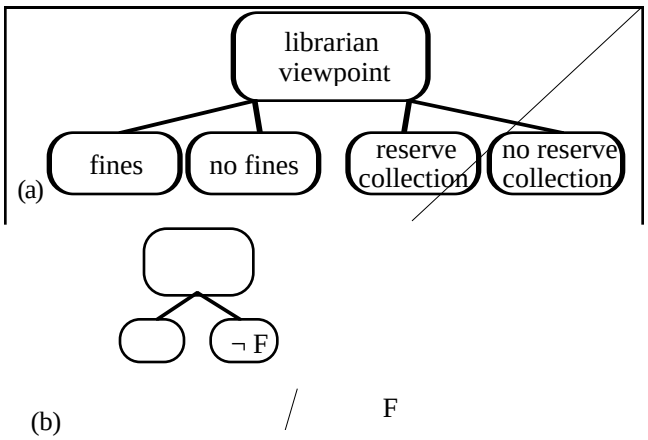
We take a 'socio-cognitive' stance, by which we mean we are interested in the interaction of cognitive and social activities, including issues of shared understanding, and the relationship of mental representations with their social and cultural settings. Instead of modelling a single problem solving agent, we are modelling an organisation. Knowledge needs to be elicited from many different sources, and hence we need to deal with many different viewpoints, and the inevitable conflicts between them. In effect, our domain model will encompass a number of different conceptual models, representing different viewpoints and different roles within an organisation.

2 Related Work

The need to deal with multiple views is central to requirements engineering, and a number of approaches


maximise(circulation)
maximise(circulation) -> lending_l
lending_limits -> fines
(a)
maximise(circulation)
maximise(circulation) -> lending_l
lending_limits -> fines
not(fines)
(b)
(c)


Librarian
Viewpoint
Description


librarian viewpoint
fines
no fines
reserve collection
no reserve collection
(a)

¬ F

(b)

F

no

Inheritance Rules for New Descendants

1) If no previous descendants exist, the usual algorithm (figure 2) is used (a, b).

2) If the item is inconsistent with the viewpoint, then it follows that it is inconsistent with all descendants. In this case, two new descendants, **X** and **Y** are created. **X** will contain the new item, while **Y** represents the status quo. Any previous descendants become descendants of **Y** (c).

3) If the item is consistent with the viewpoint, it might be inconsistent with some existing descendants. For each family, test whether the new item is consistent with each descendant. The following situations are possible:

i) The new item is consistent for all existing descendants. In this case it can be added directly to the original viewpoint (d).

ii) The new item is inconsistent with all existing descendants. In this case rule 2 above applies (c)

iii) The new item is consistent with some descendants and not others. If only one descendant in each pair is inconsistent with the new item, it is placed in the alternative to this descendant (e, f). Otherwise, two new descendants, **X** and **Y** are created, as in rule 2. Pairs that are consistent with the new item become descendants of **X**

transition diagrams. The predicate calculus is supported with a simplified set of rules for detecting conflicts, by