Abstract

An overview of a planning system for chess called N.N. is given. Our special view of planning as applying flexible action scripts to given situations and the low level planning mechanisms for concretizing these scripts is explained. The separation of plan knowledge and world knowledge is pointed out and the interaction between these knowledge sources is shown by describing the behavior of the system.

Introduction

We will present a system (called N.N.) that deals with certain aspects of planning. This system is under development with chess as its area of application. Hereby the design goal of the system is not to build a program playing full games but rather the development and examination of certain methods allowing to deal with those aspects of planning. In this approach we primarily investigate situation-dependent behaviour by applying knowledge about the typical procedure in certain situations. Therefore chess will be looked at under some restriction which may be characterized as follows:

Only quiet middle game positions will be considered, assuming that a given position belongs to a certain type of positions. Every such type of positions is associated with a prototypical procedure representing knowledge about the typical actions in this type of positions. This knowledge about possible actions is formulated by stating either atomic or composed actions of highly heuristic nature. This knowledge may be regarded as a collection of hints like

'action A may be realized by undertaking action B, if B doesn't work action C can be tried or even the actions D and E.'

All those partial actions may be provided with situation dependent conditions for their applicability. (*)

The classification of a given position is performed by looking at the pawn structure in this position. Hereby we distinguish between pawn skeletons defined by functional relations between certain pawns and fixed pawn structures defining constant stereotypes. Pawn structures defined by functional relations therefore yield a less specific classification than fixed pawn structures which by that can be associated with a more specific procedure.

Moreover, underlying the system is a model of its domain containing knowledge about all entities to be found in this domain as well as their characterization. So, for example, the entity "PAWN" is defined by its legal moves, its captures, its value etc. in a given position.

(*) Unlike e.g. /Bratko 82/ the success even of generated plans cannot be proved, at most they may be feasible. The knowledge about actions, on the other hand, is more structured than in /Kaindl 82/ and is therefore closer to the "human window" /Michie 82/.
The design goal is a system provided with knowledge which on all levels is explicit, inspectable and editable. A sketch of its dynamic behaviour, considerably simplified, is shown in Fig. 1.

Fig. 1 Simplified representation of N.N.'s dynamic behaviour

1. The Knowledge Bases

The structures of the particular knowledge bases will now be described in more detail. Their interaction and the system's dynamic behaviour will be illustrated. First the representation of N.N.'s model is sketched because the other knowledge sources to a high degree work with the information supplied by it.

1.1 Model World

The model underlying N.N. consists of some few entities representing the particular types of pieces as well as of one entity having the properties common to all piece types.

A piece type is essentially defined by its legal operations in an actual position. These operations are mainly the moves a piece of this type can make on an otherwise empty board and in an actual position as well as its possible captures. Besides that, the model derives some properties from these, like all those moves of the piece allowing it to control a given set of squares. These piece types are represented as Frames, their properties as attached procedures (compare e.g. /Stefik 79/).

By that, an actual position can be represented by describing the particular pieces which only carry information about their colour and square and a link to their respective types.

As every position defines its own frame-system more than one position at a time can be held. N.N. uses this possibility to plan in hypothetical positions by generating the successors of a given position.(*)

(*) For the sake of efficiency for special applications, like a capture analysis, a more compact coding of positions is generated, in accordance to those in a brute-force program (see /Frey 77/).