O. INTRODUCTION

We redefine here (cf. [Hab 84]) the class DPL of the Dynamic Process Logics which form a generalization of the logics from [Pra 79, Har 79]. DPL is a class of logics designated for reasoning about events during regular programs computations. For this purpose they have path formulae (interpreted over sequences of states) which describe properties of paths. The 'proper' statements of the logic are interpreted over states (state formulae). A definition of a logic \( L \) from the class DPL consists in giving an interpretation to operators which create (when applied to state formulae) the elementary path formulae. If the operators meet a regularity condition then \( L \) is decidable in time \( O(\exp cn^3) \). For example some logics strictly stronger than PL from [Har 79] are still decidable in that time. DPL\( ^+ \) is an extension of DPL allowing Boolean combinations of the elementary path formulae. Any DPL\( ^+ \)-logic meeting the same regularity condition is decidable in time \( O(\exp(n^3 \exp ck)) \) where \( k \) is a number appreciable less than the length \( n \) of the given formula. Any DPL\( ^+ \)-logic with zero-ary regular operators remain decidable in contrast to the non-local PL from [HKP 82] which is known to be undecidable. At the end we list some open problems.

1. DEFINITION OF DPL\( ^+ \) AND DPL

Assume we have given three countable, pairwise disjoint sets of actions: \( A_0, A_1, \ldots \) propositions: \( P_0, P_1, \ldots \) and operators: \( O_0, O_1, \ldots \). Programs are defined exactly as in the Propositional Dynamic Logic [FL 79] with tests over state formulae, see below.
State- and path formulae are built accordingly to the following rules:
F1. any proposition is a state formula
F2. if \( a \) is a program and \( p \) is a path formula then \( \langle a \rangle p \) is a state formula (called a "diament" formula)
F3. state formulae are closed under the Boolean connectives
F4. if \( O \) is an operator of arity \( m \) and \( p_1, \ldots, p_m \) are state formulae then \( O(p_1, \ldots, p_m) \) is an (elementary) path formula
F5. path formulae are closed under the Boolean connectives

The set of \( \text{DPL}^+ \) (DPL) -formulae consists of all the state formulae defined by F4 - F5 (F4 - F4 resp.).

Let \( S \) be a nonempty set. By \( S \) we denote the set of all (finite or not) nonempty sequences of elements from \( S \). A structure for \( \text{DPL}^+ \) is any triple \( (S, \sim, \text{Tr}) \) where \( S \) is a nonempty set of states, \( \sim \) is a relation such that \( \sim \subseteq S \times \{\text{state formulae}\} \cup S \times \{\text{path formulae}\} \), \( \text{Tr}: \{\text{programs}\} \rightarrow \text{Powerset}(S) \). A structure is a model iff it fulfills the following semantic conditions:

S1. \( \text{Tr}(a; b) = \{(s_0, \ldots, s_i, \ldots) | (s_0, \ldots, s_i) \in \text{Tr}(a) \text{ and } (s_i, \ldots, s_j, \ldots) \in \text{Tr}(b) \text{ for some } i \geq 0\} \)
S2. \( \text{Tr}(a \cup b) = \text{Tr}(a) \cup \text{Tr}(b) \)
S3. \( \text{Tr}(a^*) = \{(s) | s \in S \} \cup \bigcup_n \text{Tr}(a^n), n \geq 1 \)
S4. \( \text{Tr}(p?) = \{(a, s) | s \models p\} \)
S5. \( s \models <a)p \) iff there is \( g \in \text{Tr}(a) \) such that \( g \models p \) and the first element of \( g \) is \( s \)
S6. describe the standard behaviour of \( \models \) on the Boolean connectives

Example

In order to fix a logic in the \( \text{DPL}^+ \)-framework we have to define the operators. Let \( s = (s_0, \ldots, s_i, \ldots) \) be a path in a structure. We write simply \( p \) until \( q \) instead of \( \text{until}(p, q) \) etc.

\[ s \models p \text{ until } q \text{ iff } \forall i : s_i = q \text{ and } \exists j : i \leq j \text{ and } s_j = p \]
\[ s \models p \text{ while } q \text{ iff } \forall i : (\forall j : i \leq j \text{ and } s_j = q) \implies s_i = p \]
\[ s \models \text{next } p \text{ iff } \exists s_4 \text{ on } g \text{ and } s_4 = p \]
\[ s \models p \text{ since } q \text{ iff } \exists i : s_i = q \text{ and } \forall j : i \leq j \text{ and } s_j = p \]
\[ s \models p \text{ imp } q \text{ iff } \forall i : s_i = p \implies \exists j : i \leq j \text{ and } s_j = q \]
\[ s \models \text{ even } p \text{ iff } \forall i : s_{2i} = p \]

Using this operators some other constructs may be defined:

some \( p \) is simply true until \( p \), all \( p \) corresponds to \( p \) while true and last \( p \) to false since \( p \). Thus Pratt's "during" is expressible in \( \text{DPL}^+ \) as \( [\text{some } p \text{ and in } \text{DPL} \text{ as } \neg <a> \text{all } p \). Harel's \( \psi \)-formula is expressible in \( \text{DPL}^+ \) as \( [\text{some } p \text{ imp } q \text{ or as } \neg <a>(\neg q \text{ since } p) \) in \( \text{DPL} \), and is not definable in \( \text{FL} \) from [Har 79].

2. DECIDABILITY CRITERION

In this chapter we restrict the semantics of programs to sets of finite path only. Let \( D \subseteq \text{Subformulae}(p) \cup \{-q | q \in \text{Subformulae}(p)\} \)