Introduction.

There are many kinds and variants of logics designated for reasoning about programs. We introduce here the Dynamic Process Logic, DPL, which is a framework for logics designated for reasoning about events during regular programs computations. For this purpose they have path-formulae (interpreted over sequences of states) which tell us something about events along the path considered. The "proper" formulae are state formulae, interpreted over states. A criterion for decidability of the satisfiability problem is given, due to which any logic definable in DPL-framework and meeting a regularity condition is decidable in time $O(\exp cn^3)$. For instance some logics strictly stronger than PL from [H] 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 (usually appreciable) less than the length $n$ of the given formula.

Temporal Process Logic, TPL, is a class of temporal branching time logics which is similar to DPL without the program connectives but with infinite sequences of states as the semantics of the atomic programs. A modification of the regularity condition gives the second criterion: any TPL-logic which fulfills the second condition is decidable in $O(\exp cn^2)$-time and any TPL$^+$-logic in time $O(\exp(n^2 \exp ck))$. This is an improvement of the theorem 8.5 from [EH], where $O(\exp^2(cn - \log n))$ upper bound is given for CTL$^+$, the Computation Tree Logic with boolean combinations of the path formulae. Cf. [W] for similar results in the linear time temporal logic.
1. Tools from the automata theory

Besides the usual finite automata accepting finite strings we will also use the Büchi automata on infinite strings. We review below their definition.

Definition 1.1
\[ \alpha = (S, M, s_0, F) \] is a (deterministic) Büchi automaton on infinite strings from \( \Sigma^\omega \) iff \( S \) is a finite set of states, \( s_0 \in S \) is the initial state, \( F \subseteq S \) and \( M : S \times \Sigma \rightarrow S \). Let \( w = \sigma_0 \sigma_1 \ldots \) be an \( \omega \)-sequence from \( \Sigma^\omega \). A run of \( \alpha \) on \( w \) is the function \( r : \omega \rightarrow S \) such that \( r(0) = s_0 \) and \( r(i+1) = M(r(i), \sigma_i) \) for any natural \( i \). \( \alpha \) accepts \( w \) iff \( \{ s \mid r(i) = s \text{ for infinitely many } i \} \cap F \neq \emptyset \), for the unique run \( r \) of \( \alpha \) on \( w \).

Automata on trees are of crucial importance for our decision procedure:

Definition 1.2
The \( N \)-ary infinite tree \( T_N \), \( N \geq 1 \), is the set \{ 0, \ldots, N-1 \} \( \times \) \( \Sigma^\omega \). A path \( (x_0, x_1, x_2, \ldots) \) in \( T_N \) is any (finite or not) sequence of nodes from \( T_N \) such that \( x_{i+1} \) is an immediate successor of \( x_i \). A finite \( N \)-ary tree \( T \) is any finite subset of \( T_N \) such that:
- the empty string, root of \( T_N \), belongs to \( T \) and
- if \( x \in T \) then no successor of \( x \) in \( T_N \) belongs to \( T \) or all the immediate successors belong to \( T \).

By \( Fr(T) \), frontier of \( T \), we denote \{ \( x \in T \mid x \) has no successors in \( T \} \). A \( N \)-ary infinite (finite resp.) \( \Sigma \) -tree is any function \( f : T_N \rightarrow \Sigma \) ( \( f : T \rightarrow \Sigma \) for some finite tree \( T \subseteq T_N \)).

Now we fix \( N \) and omit it in subsequent definitions.

Definition 1.3
A special automaton on (infinite) \( \Sigma \) -trees, see [R], is a tuple \( \text{Aut} = (S, M, s_0, F) \) where \( S, s_0 \) and \( F \) are defined as for Büchi automata and \( M : S \times \Sigma \rightarrow \text{Powerset}(S) \). \( r \) is a run of \( \text{Aut} \) on \( f : T_N \rightarrow \Sigma \) iff \( r : T_N \rightarrow S \) is such that:
- \( r(\Lambda) = s_0 \), \( \Lambda \) is the root of \( T_N \), and
- \( (r(x_0), \ldots, r(x(N-1))) \in M(r(x), f(x)) \) for any \( x \) in \( T_N \).

\( \text{Aut} \) accepts \( f : T_N \rightarrow \Sigma \) iff there is a run \( r \) of \( \text{Aut} \) on