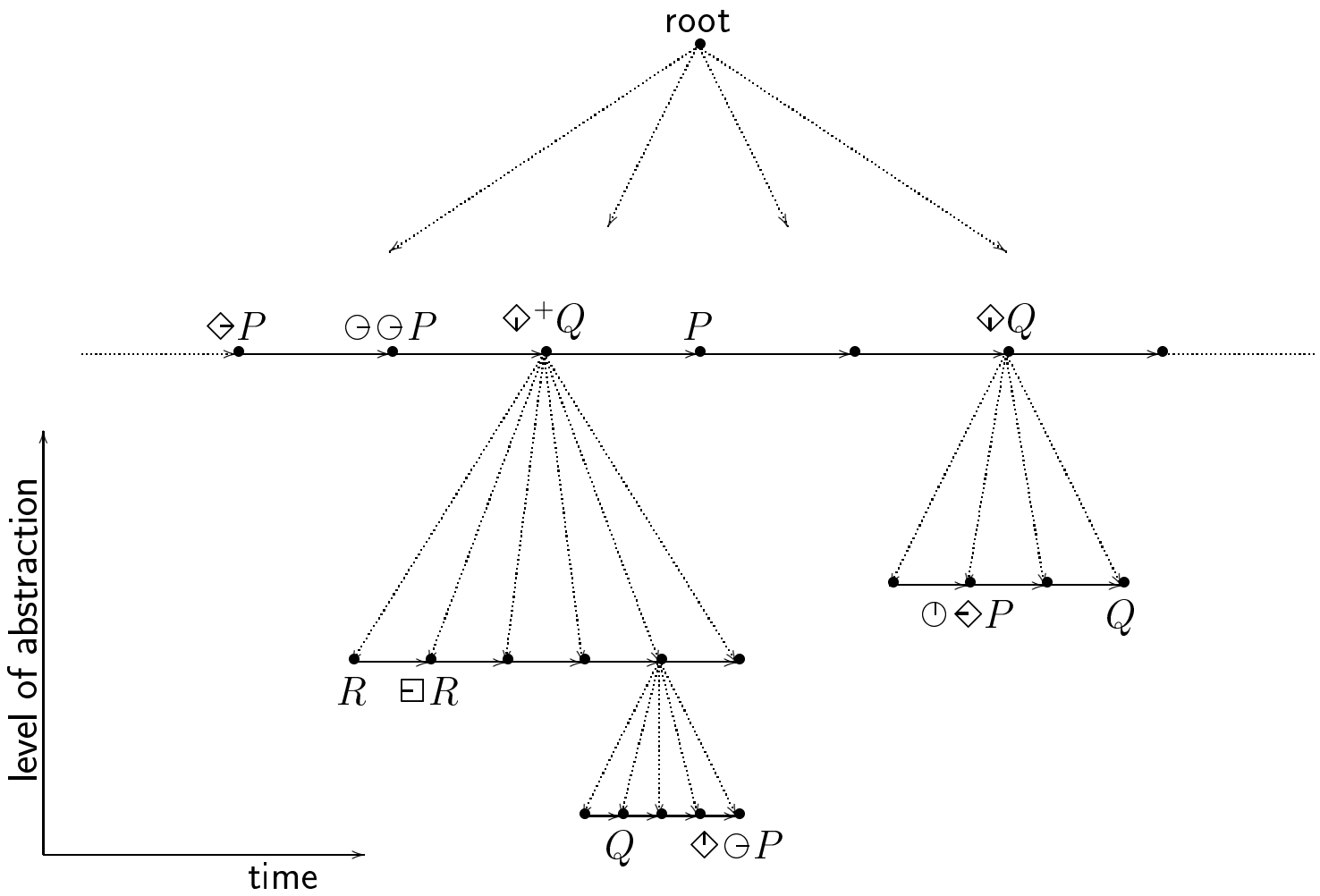
Ulle Endriss

Department of Computer Science, King's College London
Strand, London WC2R 2LS, United Kingdom
Email: endriss@dcs.kcl.ac.uk

Motivation. Linear temporal logic has been applied successfully to problems in program specification and verification. However, this logic does not support the notion of refinement in a natural manner.

We propose to overcome this shortcoming by *adding a zoom to linear temporal logic*. The result is a modal logic with models that are based on *ordered trees*.



The figure illustrates the move from linear time to ordered trees. The horizontal modalities are taken from temporal logic: $\diamondsuit$ "sometime in the future", $\boxminus$ "always in the future", $\ominus$ "in the next state" (and analogously for the past).

Syntax. The set of *well-formed formulas* A of our logic is formed according to the following BNF production rule (p stands for propositional letters):

$$A \ ::= \ p \mid \neg A \mid A \wedge A \mid \ominus A \mid \ominus A \mid \circledcirc A \mid \diamondleft A \mid \diamondright A \mid \diamondsuit A \mid \diamondsuit^+ A$$

Additional propositional connectives and box-operators may be introduced as defined operators in the usual way (like, for example, $\boxminus\varphi = \neg\diamondright\neg\varphi$).

Semantics. Our logic is a multi-modal logic over frames that are ordered trees:

▶ An *ordered tree* is a tree where the children of each node form a *strict linear order* (Of particular interest will be the case where the children of each node are required to form a *discrete order*.)

▶ A *model* is a pair $\mathcal{M} = (\mathcal{T}, V)$, where $\mathcal{T}$ is an ordered tree and V is a valuation function mapping propositional letters to sets of nodes of $\mathcal{T}$.

▶ We inductively define the notion of *satisfaction* (or *truth*) of a formula φ in a model $\mathcal{M} = (\mathcal{T}, V)$ at a node t within $\mathcal{T}$ as follows:

1. $\mathcal{M}, t \models p$ iff $t \in V(p)$ for propositional letters p
2. $\mathcal{M}, t \models \neg\varphi$ iff not $\mathcal{M}, t \models \varphi$
3. $\mathcal{M}, t \models \varphi \wedge \psi$ iff $\mathcal{M}, t \models \varphi$ and $\mathcal{M}, t \models \psi$
4. $\mathcal{M}, t \models \circledcirc\varphi$ iff t has a parent t' (i.e. t is not the root) and $\mathcal{M}, t' \models \varphi$
5. $\mathcal{M}, t \models \ominus\varphi$ iff t has a direct left neighbour t' and $\mathcal{M}, t' \models \varphi$
6. $\mathcal{M}, t \models \ominus\varphi$ iff t has a direct right neighbour t' and $\mathcal{M}, t' \models \varphi$
7. $\mathcal{M}, t \models \diamondsuit\varphi$ iff t has an ancestor t' with $\mathcal{M}, t' \models \varphi$
8. $\mathcal{M}, t \models \diamondleft\varphi$ iff t has a left sibling t' with $\mathcal{M}, t' \models \varphi$
9. $\mathcal{M}, t \models \diamondright\varphi$ iff t has a right sibling t' with $\mathcal{M}, t' \models \varphi$
10. $\mathcal{M}, t \models \diamondsuit\varphi$ iff t has a child t' with $\mathcal{M}, t' \models \varphi$
11. $\mathcal{M}, t \models \diamondsuit^+\varphi$ iff t has a descendant t' with $\mathcal{M}, t' \models \varphi$

Satisfiability and validity. A formula φ is called *satisfiable* iff it has a model (i.e. iff there are a model $\mathcal{M} = (\mathcal{T}, V)$ and a node t within $\mathcal{T}$ with $\mathcal{M}, t \models \varphi$). A formula φ is called *valid* (in an ordered tree $\mathcal{T}$) iff it is satisfied at every node in every model (based on $\mathcal{T}$).

Reflexive modalities. We give a number of examples to demonstrate the expressive power of ordered tree logics. Some of these will make use of the following reflexive variants of the transitive modalities available in our logic:

$$\begin{array}{lcl@{\qquad}lcl} \Diamond^*\varphi & = & \varphi \vee \Diamond\varphi & \Box^*\varphi & = & \varphi \wedge \Box\varphi \\ \Diamond\!\!\!\!-^*\varphi & = & \varphi \vee \Diamond\!\!\!\!-\varphi & \Box\!\!\!\!-^*\varphi & = & \varphi \wedge \Box\!\!\!\!-\varphi \\ \Diamond\!\!\!\!\rightarrow^*\varphi & = & \varphi \vee \Diamond\!\!\!\!\rightarrow\varphi & \Box\!\!\!\!\rightarrow^*\varphi & = & \varphi \wedge \Box\!\!\!\!\rightarrow\varphi \\ \Diamond_\downarrow^*\varphi & = & \varphi \vee \Diamond_\downarrow^+\varphi & \Box_\downarrow^*\varphi & = & \varphi \wedge \Box_\downarrow^+\varphi \end{array}$$

Universal modality. The universal modality is definable in our logic. The following two operators range over all nodes in a tree:

$$\Diamond\varphi \;=\; \Diamond^*\Diamond_\downarrow^*\varphi \qquad\qquad \Box\varphi \;=\; \Box^*\Box_\downarrow^*\varphi$$

Tree features. We can easily characterise simple features of nodes in a tree, such as being the root or being a leftmost child:

$$\begin{array}{rcl@{\qquad}rcl} \textsc{root} & = & \neg\,\Diamond\!\!\!\!\leftarrow\top & \textsc{leftmost} & = & \neg\,\Diamond\!\!\!\!-\top \\ \textsc{leaf} & = & \neg\,\Diamond_\downarrow\top & \textsc{rightmost} & = & \neg\,\Diamond\!\!\!\!\rightarrow\top \end{array}$$

Correspondence results. We can use axiom schemata to characterise certain classes of ordered trees. For each of the following, all instances of the given axiom schema are valid in a tree iff that tree has the corresponding property (all but the last one refer to the branching factor, that is, the orders declared over children):

$$\begin{array}{rcl} \textsc{binary} & = & \textsc{leftmost} \vee \textsc{rightmost} \\ \textsc{bounded} & = & \Diamond\!\!\!\!\leftarrow^*\textsc{leftmost} \wedge \Diamond\!\!\!\!\rightarrow^*\textsc{rightmost} \\ \textsc{dense} & = & \neg\,\Diamond\!\!\!\!-\top \wedge \neg\,\Diamond\!\!\!\!-\top \\ \textsc{weakly-discrete} & = & (\Diamond\!\!\!\!\leftarrow\top \rightarrow \Diamond\!\!\!\!-\top) \wedge (\Diamond\!\!\!\!\rightarrow\top \rightarrow \Diamond\!\!\!\!-\top) \\ \textsc{discrete} & = & \Box\!\!\!\!-(\Box\!\!\!\!-A \rightarrow A) \rightarrow (\Diamond\!\!\!\!\rightarrow\Box\!\!\!\!-A \rightarrow \Box\!\!\!\!-A) \\ \textsc{finite-branching} & = & \textsc{bounded} \wedge \textsc{discrete} \\ \textsc{finite-depth} & = & A \;\rightarrow\; \Diamond_\downarrow^*(A \wedge \Box_\downarrow^+\neg A) \end{array}$$

Note that we distinguish *(strongly) discrete branching* (between any two siblings there are only finitely many other nodes) and *weakly discrete branching* (every node with a left/right sibling must also have a closest left/right sibling).