COMP424/524-06A
Topics in Software Engineering

Part I – *CTL Model Checking*

7. Temporal Logics

Robi Malik

THE UNIVERSITY OF WAIKATO *Te Whare Wānanga o Waikato*
DEPARTMENT OF COMPUTER SCIENCE
TARI ROROHIKO

Describing Dynamic Behaviour

Example:

"The cat always returns to room 2."

$\forall t_1 : \mathbf{T} \bullet \exists t_2 : \mathbf{T} \bullet t_2 \geq t_1 \wedge \mathrm{pos}(\mathrm{cat}, t_2) = 2$

where pos(cat, t) represents the position of the cat at time t.

- First-order logic (Z-style)

Describing Dynamic Behaviour

Another example:

"The elevator never traverses a floor for which a request is pending without satisfying the request."

$$\forall t_1 : \mathbf{T} \bullet \forall t_2 : \mathbf{T} \bullet \forall\, n : \mathbb{N} \bullet$$
$$(\mathrm{req}(n, t_1) \wedge t_1 < t_2 \wedge \mathrm{pos}(t_2) \neq n \wedge$$
$$(\exists\, t_{trav} : \mathbf{T} \bullet t_1 \leq t_{trav} \leq t_2 \wedge \mathrm{pos}(t_{trav}) = n)) \Rightarrow$$
$$(\exists\, t_{serv} : \mathbf{T} \bullet t_1 \leq t_{serv} \leq t_2 \wedge \mathrm{serv}(n, t_{serv}))$$

Temporal Logic

- An extension of propositional logic.
- A more direct way of describing dynamic behaviour.
- Operators to support linguistic ways of expressing temporal relationships: **always**, **sometimes**, **until**, …

Temporal Logic

Temporal Operators:

$\mathbf{G}\, p$ "It will always be the case that p."
"p will always be true."

$\mathbf{F}\, p$ "It will sometimes be the case that p."
"p will eventually occur."

$\mathbf{X}\, p$ "p will be true in the next state."
"p will occur tomorrow (in the next step)."

Temporal Logic — Examples

$\mathbf{G}\,(\mathrm{rain} \vee \mathrm{sun})$
Every day there will be rain or sunshine.

$\mathbf{F}\,\mathrm{rain}$
It will rain some day.

$\mathbf{G}\,\neg\mathrm{rain}$
It will never rain.

$\mathbf{G}\,\mathbf{F}\,\mathrm{rain}$
Every day will be followed by a rainy day, i.e., it will rain infinitely often.

Atomic Propositions

- Propositional logic is built from atomic propositions.
- Examples:
  rain, error, floor = 3, …
- We need to extend automata to include propositions.

Kripke Structures

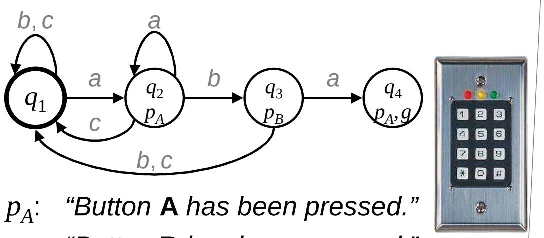
A **Kripke structure** is a tuple

$$K = (Q, Prop, T, q_0, l)$$

with

- finite set of **states** $Q = \{q_1, q_2, q_3, \dots\}$
- finite set of **propositions** $Prop = \{p_1, \dots\}$
- **transition** relation $T \subseteq Q \times Q$
- **initial** state $q_0 \in Q$
- **labelling** function $l\colon Q \to \mathbb{P}\, Prop$

Kripke Structure for Digicode



p_A: *"Button **A** has been pressed."*
p_B: *"Button **B** has been pressed."*
g: *"Passage is granted."*

Kripke Structures

Automaton
– Events
– Marked States
\+ Propositions

= Kripke-Structure

Propositional Formulas

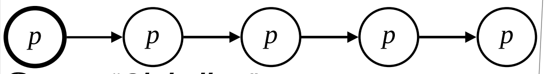
Every propositional formula is a temporal formula.

- p_A
- $p_A \vee p_B$
- $\neg(p_A \wedge p_B)$
- $p_A \Rightarrow g$
- $g \Leftrightarrow p_A$

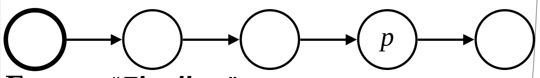
Each formula characterises a set of states.

Temporal Combinators

Temporal combinators enable us to describe a single execution sequence.

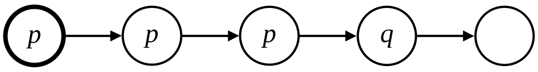


G p "***Globally*** p"



F p "***Finally*** p"

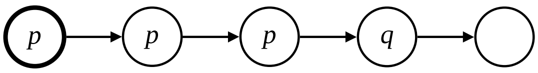
The Until Combinator



p **U** q, "p until q", is true for an execution if

- q is true at some state, and
- p is true at all states between the start state and the state where q holds.

The Weak Until Combinator

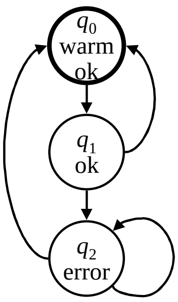


$$p \mathbf{W} q \equiv (p \mathbf{U} q) \vee \mathbf{G}p$$

"p waiting for q" or "p weak-until q"

p does not become false before a state where q holds is reached.

Examples



warm $\Rightarrow$ **F** $\neg$warm

G (warm $\Rightarrow$ **F** $\neg$warm)

G (warm $\Rightarrow$ **X** $\neg$warm)

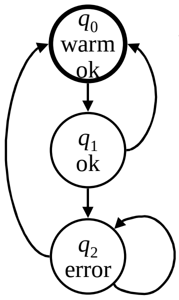
warm $\Rightarrow$ **F** error

ok **U** error

ok **W** error

warm **W** error

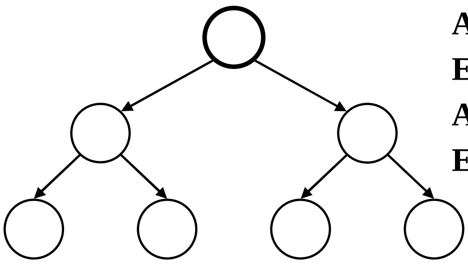
Path Quantifiers



Aφ – holds in a state from which all executions satisfy the path formula φ.

Eφ – holds in a state from which there exists an execution satisfying the path formula φ.

Path Quantifiers in CTL



AG p

EG q

AF r

EF s

Reading

Bérard et. al.:

2.1 – The Language of Temporal Logic