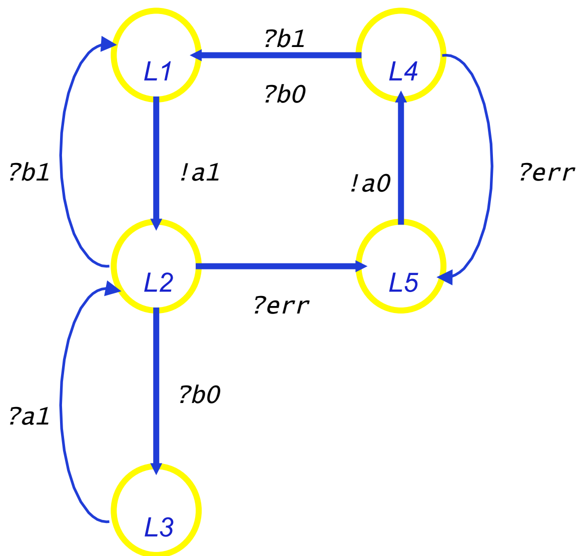
Temporal Logic
The main ideas

Ralf Möller
Hamburg University of Technology

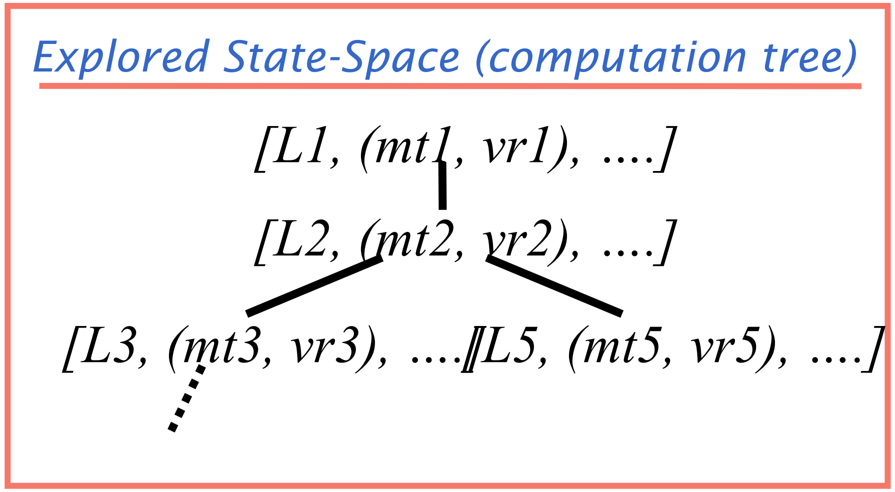
Acknowledgements

- Slides by Eric Madelaine, INRIA

Reasoning about Executions



Conceptual View



- We would like to reason about execution trees
  - tree node = snapshot of the program's state
- Reasoning consists of two layers
  - defining predicates on the program states (control points, variable values)
  - expressing temporal relationships between those predicates

Computational Tree Logic (CTL)

Syntax

Φ ::= P ...primitive propositions
| !Φ | Φ && Φ | Φ || Φ | Φ -> Φ ...propositional connectives
| AG Φ | EG Φ | AF Φ | EF Φ ...temporal operators
| AX Φ | EX Φ | A[Φ U Φ] | E[Φ U Φ]

Semantic Intuition

path quantifier

temporal operator

AG p ...along All paths p holds Globally

EG p ...there Exists a path where p holds Globally

AF p ...along All paths p holds at some state in the Future

EF p ...there Exists a path where p holds at some state in the Future

Computational Tree Logic (CTL)

Syntax

$$
\begin{aligned}
\Phi ::= \;& P && \text{...primitive propositions}\\
\mid\;& !\Phi \mid \Phi \; \&\& \; \Phi \mid \Phi \; || \; \Phi \mid \Phi \; \text{->} \; \Phi && \text{...propositional connectives}\\
\mid\;& AG\ \Phi \mid EG\ \Phi \mid AF\ \Phi \mid EF\ \Phi && \text{...path/temporal operators}\\
\mid\;& AX\ \Phi \mid EX\ \Phi \mid A[\Phi\ U\ \Phi] \mid E[\Phi\ U\ \Phi]
\end{aligned}
$$

Semantic Intuition

AX p …along **A**ll paths, p holds in the ne**X**t state

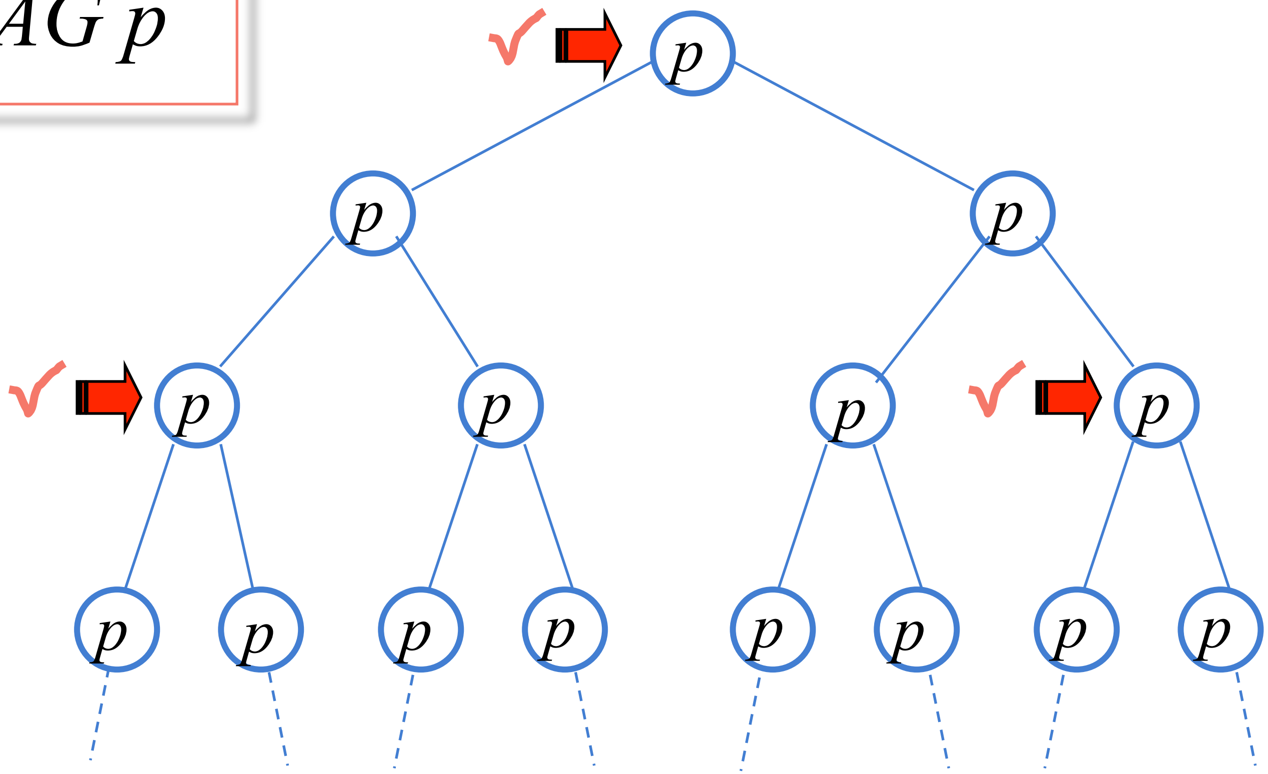
EX p …there **E**xists a path where p holds in the ne**X**t state

A[p U q] …along **A**ll paths, p holds **U**ntil q holds

E[p U q] …there **E**xists a path where p holds **U**ntil q holds

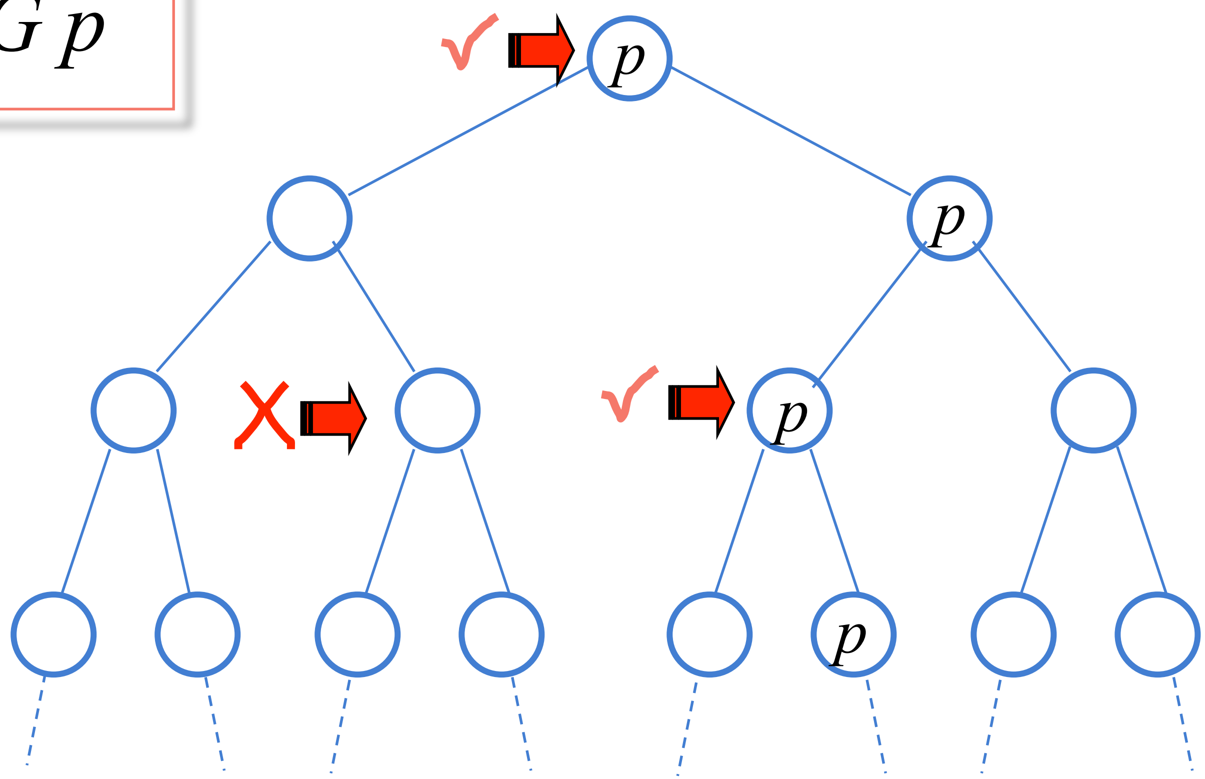
Computation Tree Logic

$AG\, p$



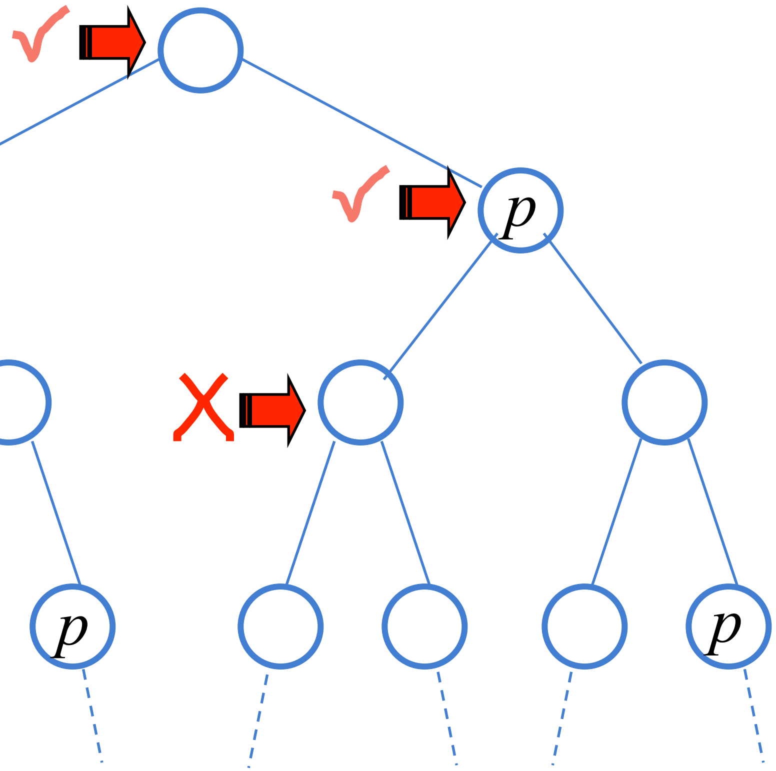
Computation Tree Logic

$EG\ p$



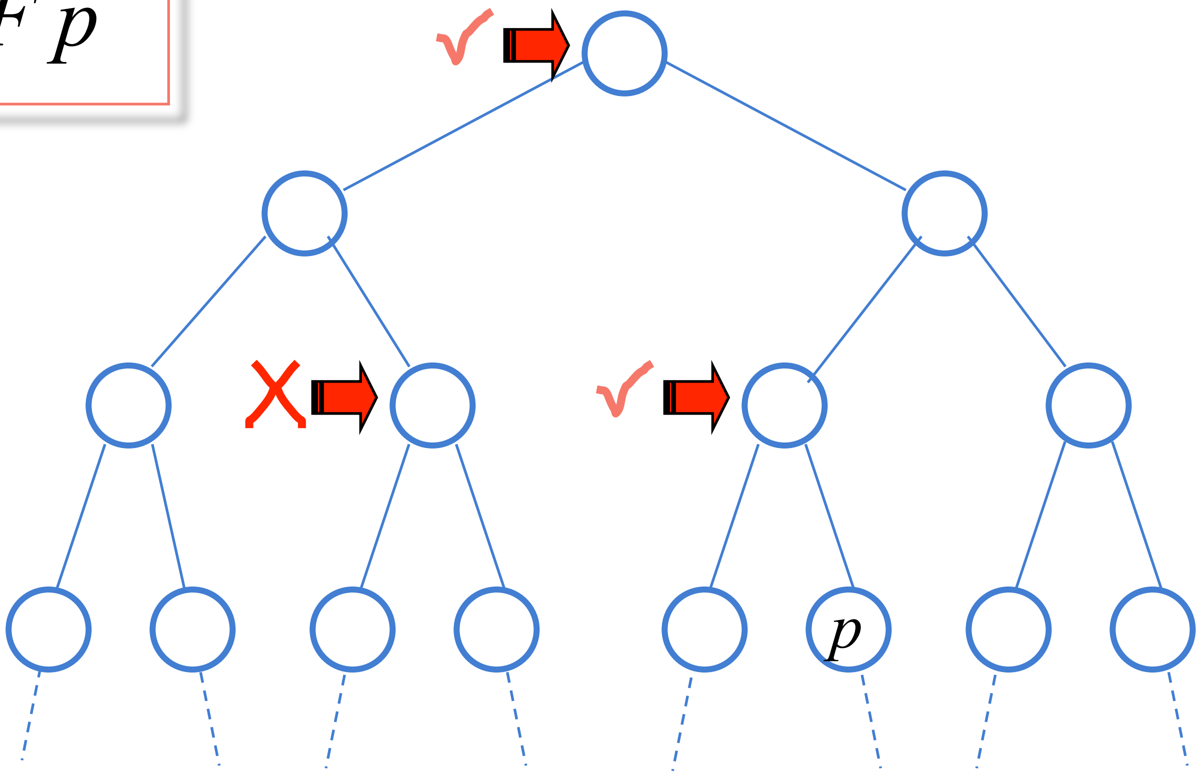
Computation Tree Logic

$AF\ p$



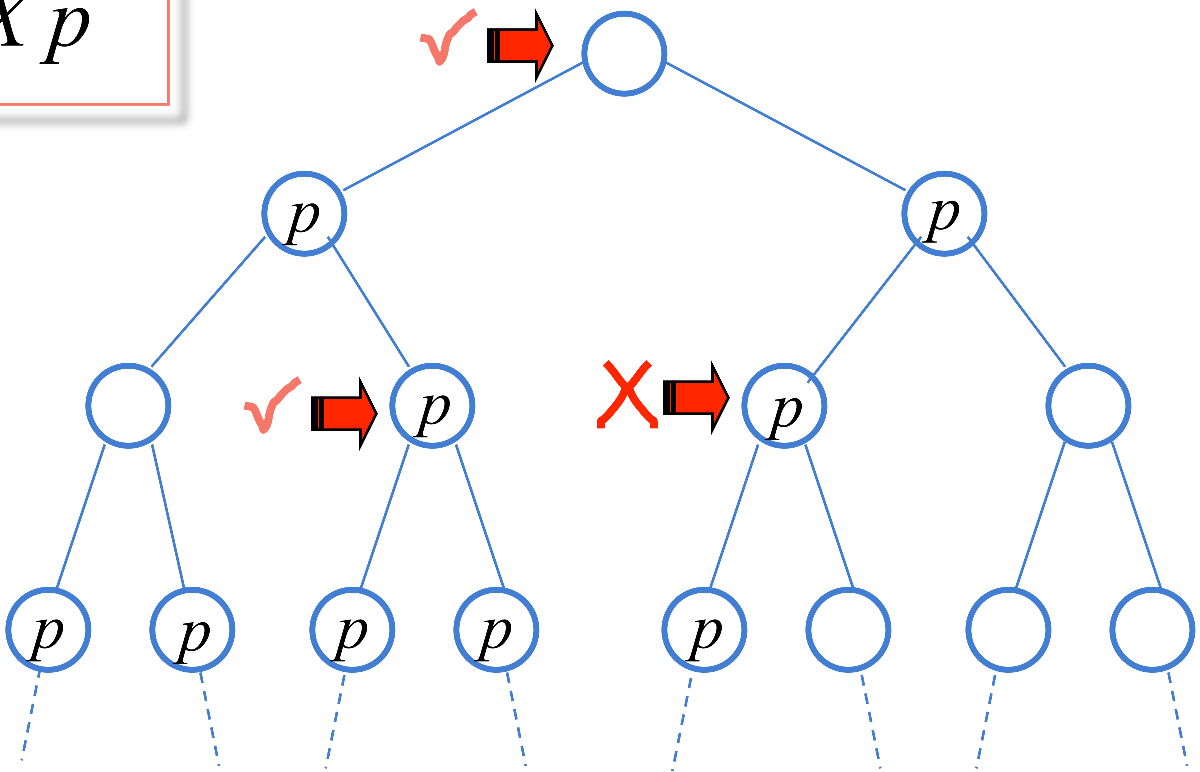
Computation Tree Logic

EF p



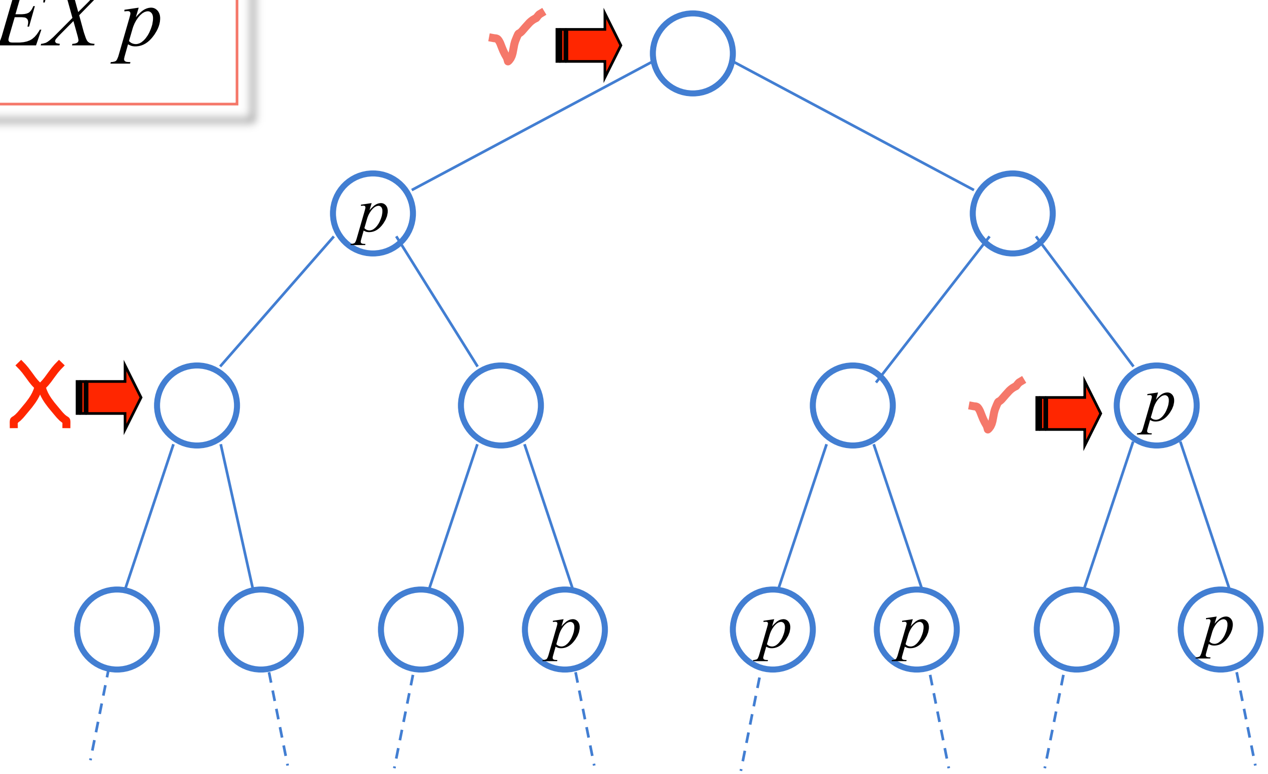
Computation Tree Logic

$AX\,p$



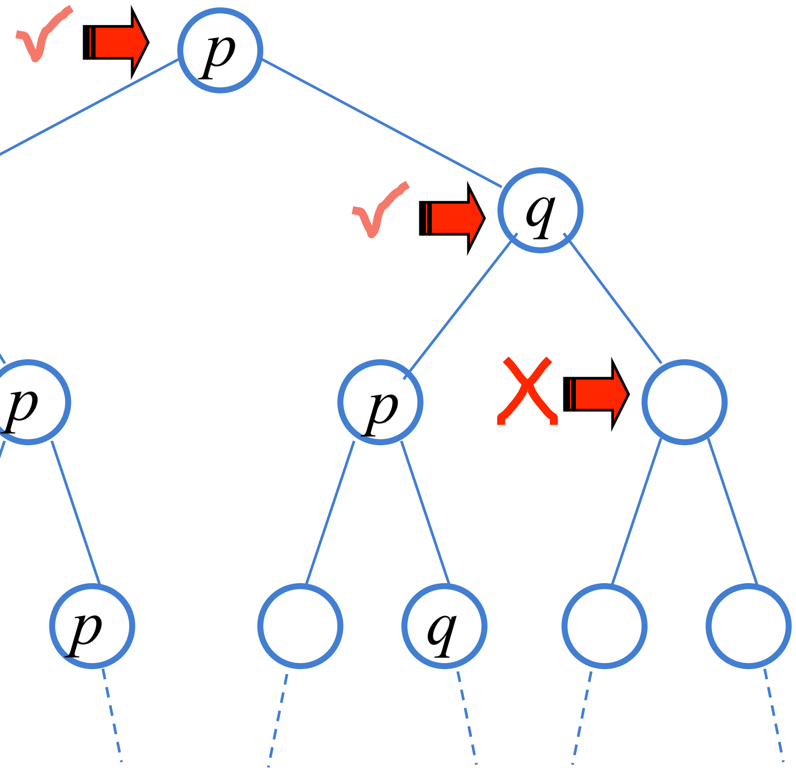
Computation Tree Logic

$EX\,p$



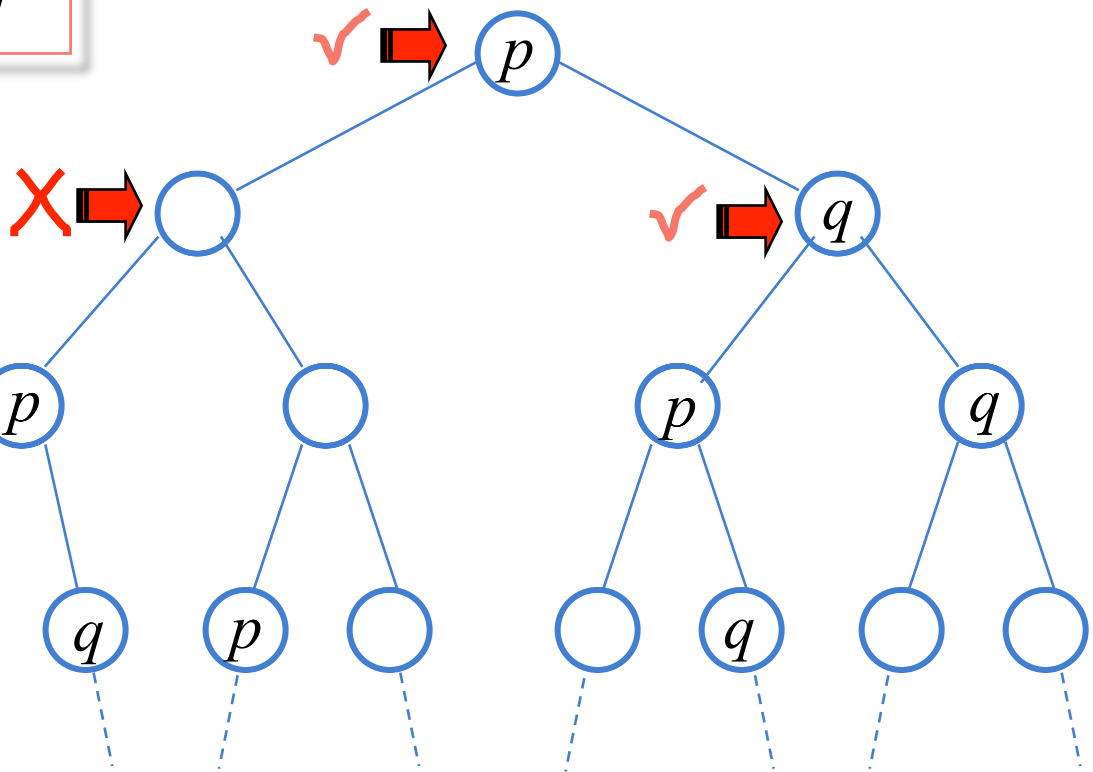
Computation Tree Logic

A[p U q]



Computation Tree Logic

E[p U q]



Example CTL Specifications

- For any state, a request (for some resource) will eventually be acknowledged

AG(requested -> AF acknowledged)

- *From any state, it is possible to get to a restart state*

AG(EF restart)

- *An upwards travelling elevator at the second floor does not changes its direction when it has passengers waiting to go to the fifth floor*

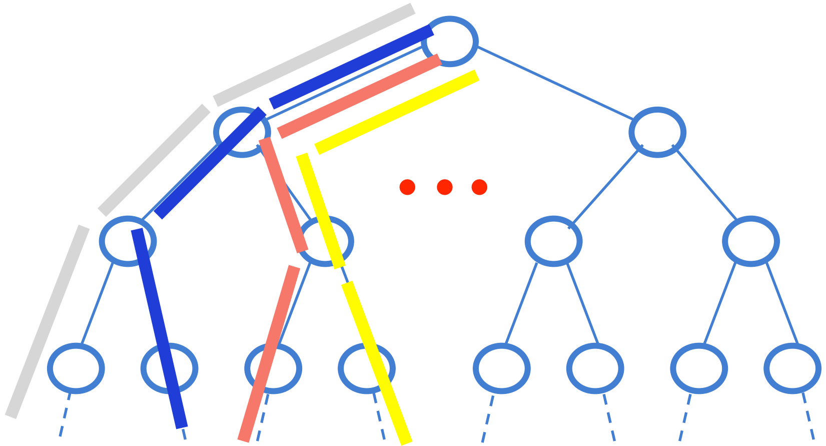
AG((floor=2 && direction=up && button5pressed)
-> A[direction=up U floor=5])

CTL Notes

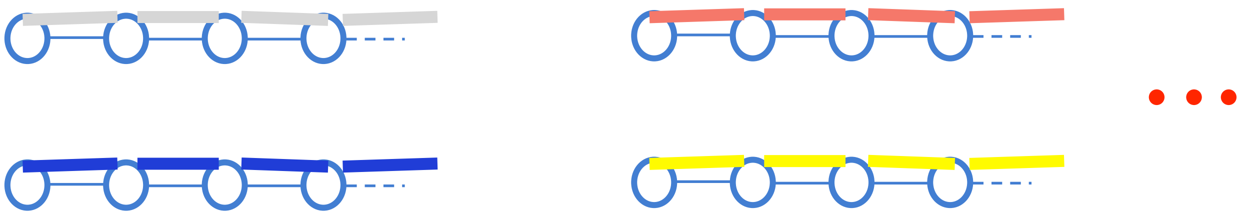
- Invented by E. Clarke and E. A. Emerson (early 1980's)
- Specification language for Symbolic Model Verifier (SMV) model-checker
- SMV is a *symbolic* model-checker instead of an *explicit-state* model-checker
- Symbolic model-checking uses Binary Decision Diagrams (BDDs) to represent boolean functions (both transition system and specification

Linear Temporal Logic

Restrict path quantification to "ALL" *(no* "EXISTS"*)*



Reason in terms of linear traces *instead of branching* trees



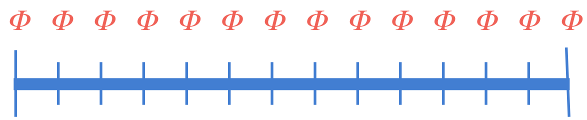
Linear Temporal Logic (LTL)

Syntax

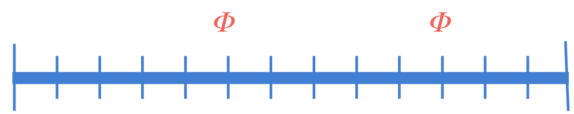
Φ ::= P ...primitive propositions
| !Φ | Φ && Φ | Φ || Φ | Φ -> Φ ...propositional connectives
| []Φ | <>Φ | Φ U Φ | X Φ ...temporal operators

Semantic Intuition

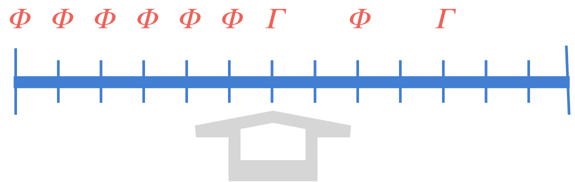
[]Φ ...always Φ



<>Φ ...eventually Φ



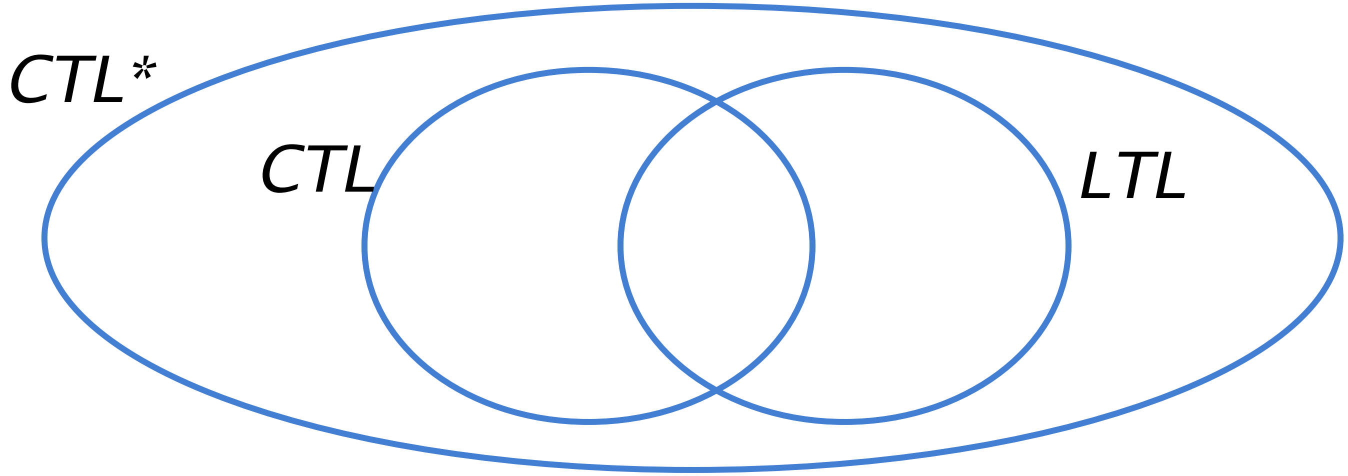
Φ U Γ ...Φ until Γ



LTL Notes

- Invented by Prior (1960's), and first use to reason about concurrent systems by A. Pnueli, Z. Manna, etc.
- LTL model-checkers are usually explicit-state checkers due to connection between LTL and automata theory
- Most popular LTL-based checker is Spin (G. Holzman)

Comparing LTL and CTL



- *CTL is not strictly more expressive than LTL (and vice versa)*
- *CTL* invented by Emerson and Halpern in 1986 to unify CTL and LTL*