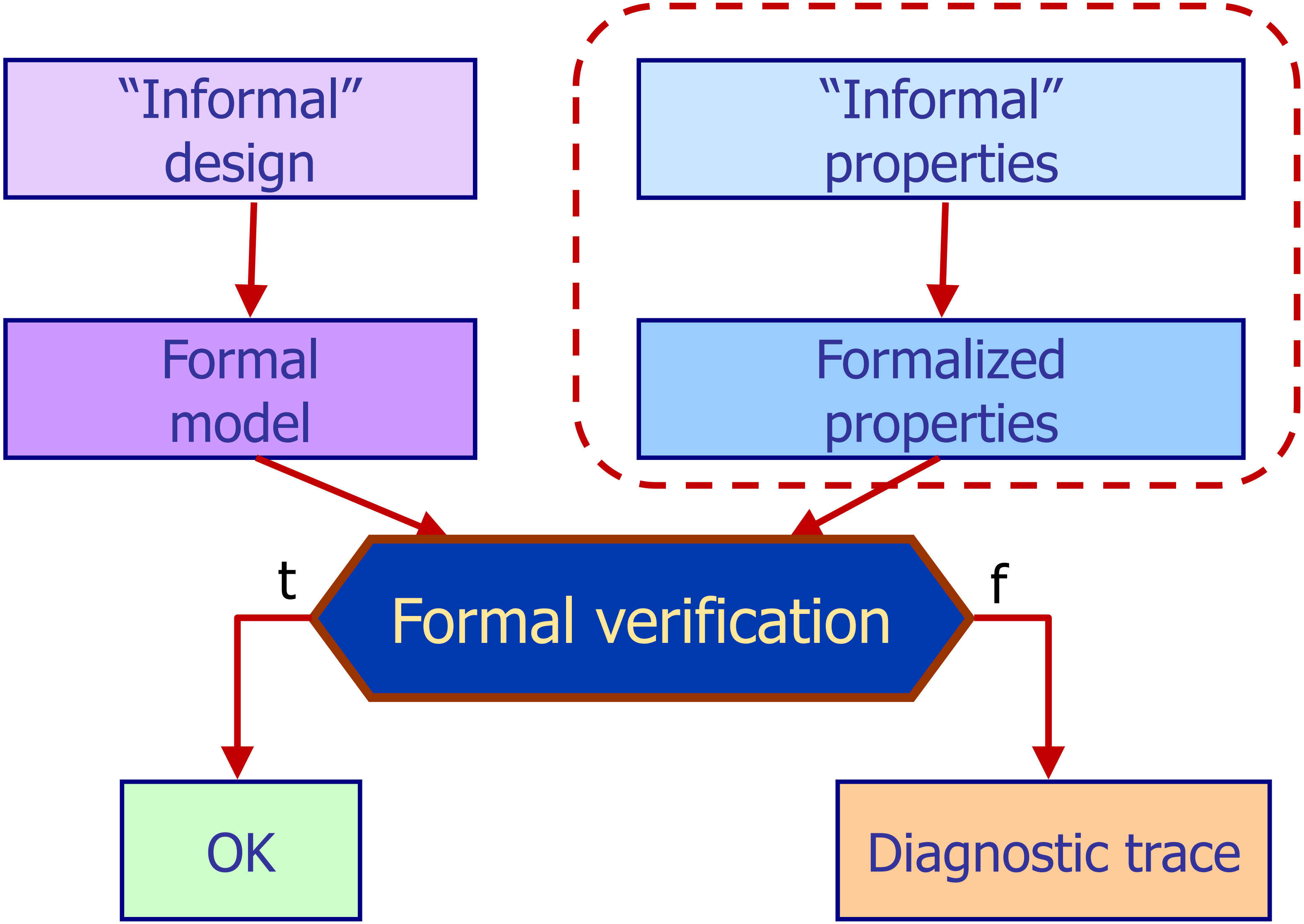
Formalizing and checking properties: Temporal logics

Istvan Majzik
majzik@mit.bme.hu

Budapest University of Technology and Economics
Dept. of Measurement and Information Systems

Recap: Goals of formal verification



Overview

- Formalization of requirements
  - Frequent patterns
- Categories of temporal logics
  - Linear time temporal logics
  - Branching time temporal logics
- HML: Hennessy-Milner logic
  - Temporal operators
  - Model checking: Tableau method

Formalization of requirements

Frequent patterns of requirements

Handling textual requirements

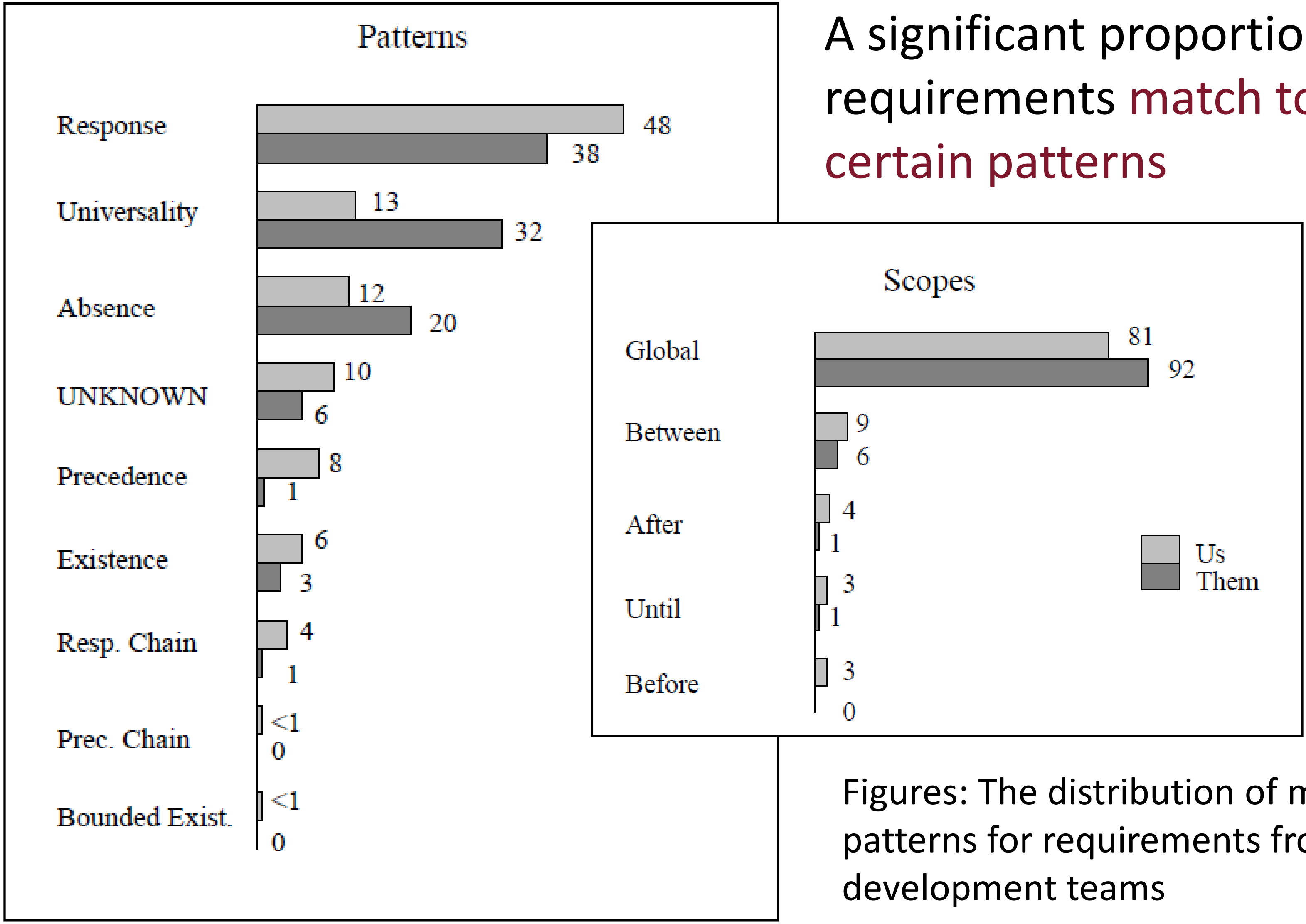
- Specifying a requirement in natural language:

> If alarm is on and alert occurs, the output of safety should be true as long as alarm is on.

> If the switch is turned to AUTO, and the light intensity is LOW then the headlights should stay or turn immediately ON, afterwards the headlights should continue to stay ON in AUTO as long as the light intensity is not HIGH.

  - Is the textual description easy to understand and unambiguous?
  - Structure is not clear (conditions, sequence, …)

Requirements in critical systems: result of a survey



A significant proportion of requirements match to certain patterns

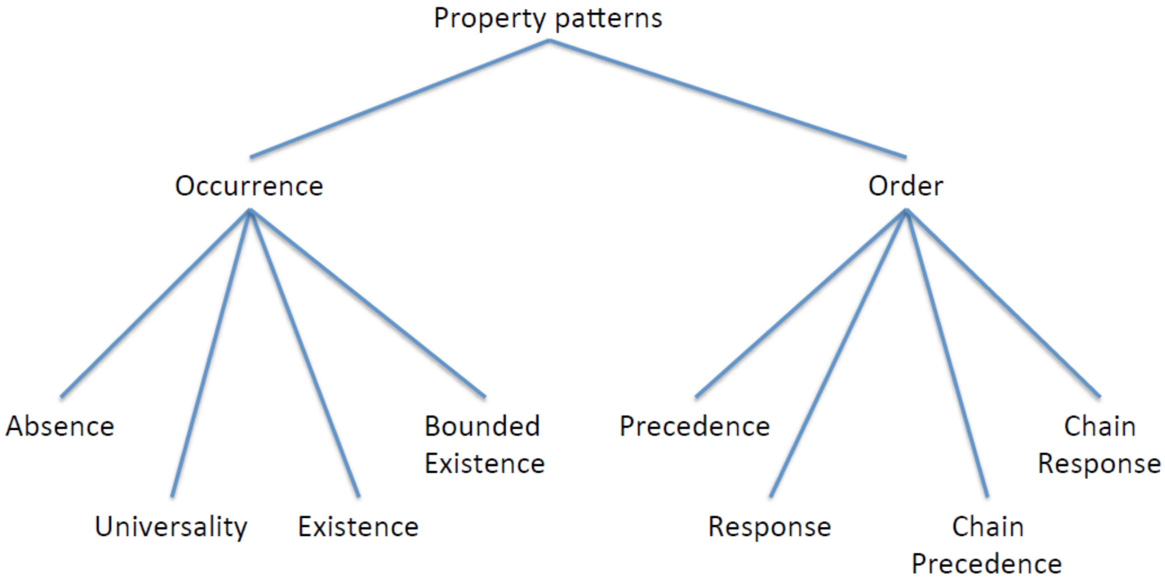
Figures: The distribution of matched patterns for requirements from two development teams

http://patterns.projects.cis.ksu.edu/documentation/patterns.shtml

Groups of patterns

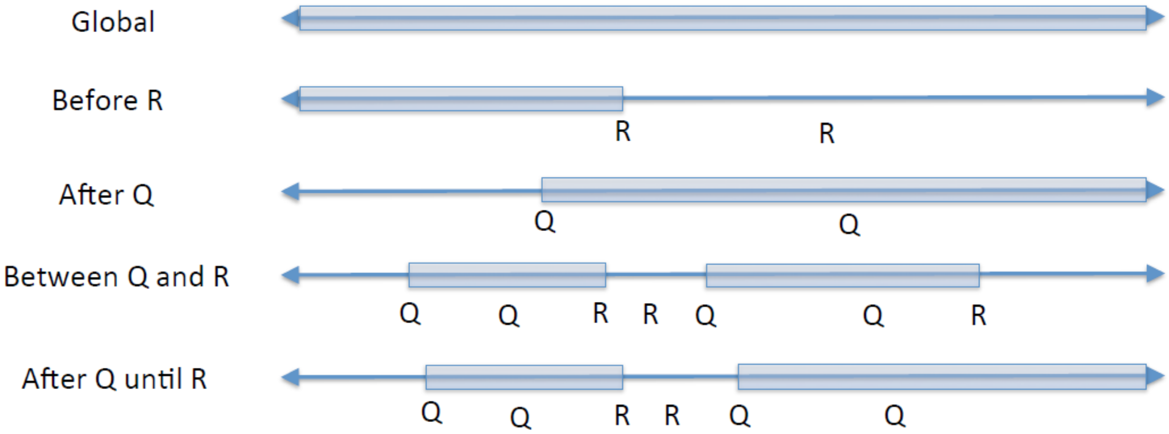
Pattern:

order or occurrence



Scope:

relative to further events



Patterns: Explanations

Occurrence:

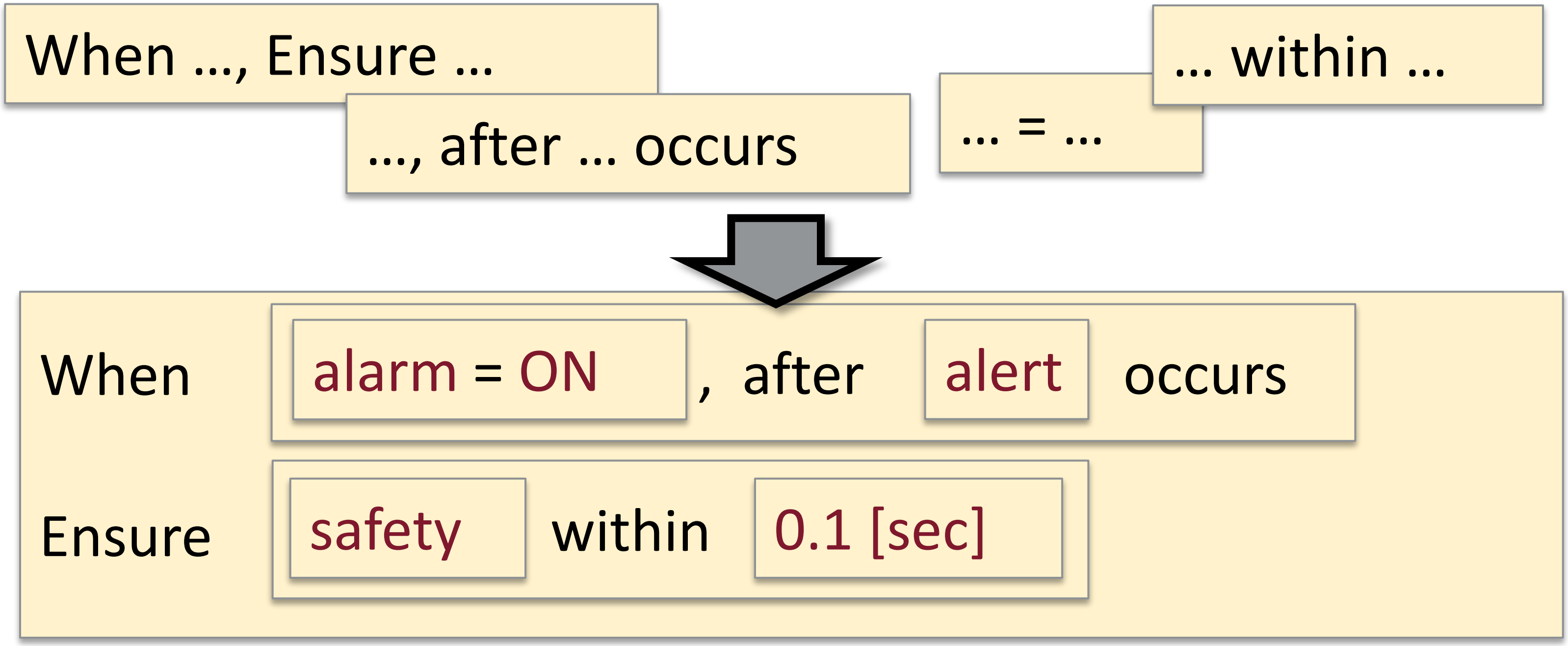
- Absence: the referenced state/event never occurs
- Universality: the referenced state/event is always present
- Existence: the referenced state/event eventually occurs
- Bounded existence: the referenced state/event occurs at least k times

Order:

- Precedence: the referenced state/event precedes another state/event
- Response: the referenced state/event is followed by another state/event
- Chain precedence: generalization of Precedence to sequences
- Chain response: generalization of Response to sequences

Composition of patterns

- Patterns can be composed
  - Boolean operators (and, or, implication)
  - Embedding patterns in other patterns
- Example: Patterns in textual form



When …, Ensure …

…, after … occurs

… = …

… within …

When alarm = ON , after alert occurs

Ensure safety within 0.1 [sec]

Outcome

- The majority of properties match certain patterns
  - If … then …, Never …, After …, Before …
  - Occurrence/order of states/events in given scope
  - More complex requirements composed from simpler ones
- These properties can be formalized if the basic patterns can be captured using a formal language
  - Absence, universality, existence, precedence, response
  - Temporal scope (globally, after, before)
- Formalization of requirements helps
  - Verification of design: exhaustive analysis of executions
  - Test evaluation, runtime monitoring: components can be automatically generated
- Applied formalism: Temporal logic

Preview: Formalization of properties in LTL

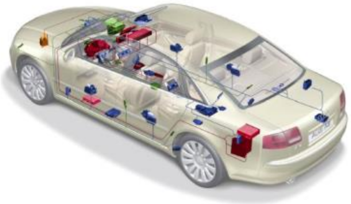
Universality within scope	**Property in LTL**	**Meaning of LTL expressions**
Event P occurs in each step of the execution globally.	$G\ P$	Globally P
Event P occurs in each step of the execution before event Q.	$F\ Q \rightarrow (P\ U\ Q)$	(Eventually Q) implies (P Until Q)
Event P occurs in each step of the execution after event Q.	$G\ (Q \rightarrow G\ P)$	Globally (Q implies Globally P)
Event P occurs in each step of the execution between events Q and R.	$G\ ((Q \wedge \neg R \wedge F\ R) \rightarrow (P\ U\ R))$	Globally ((Q and not R and Eventually R) implies (P Until R))

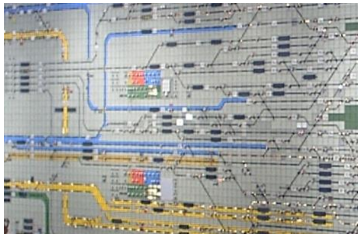
Existence within scope	**Property in LTL**
Event P occurs in the execution globally.	$F\ P$
Event P occurs in the execution before event Q.	$\neg\ Q\ W\ (P \wedge \neg\ Q)$
Event P occurs in the execution after event Q.	$G\ (\neg Q) \vee F\ (Q \wedge F\ P))$
Event P occurs in the execution between events Q and R.	$G\ (((Q \wedge \neg R) \wedge (F\ R)) \rightarrow (\neg R\ W\ (P \wedge \neg R)))$

Temporal requirements and temporal logics

Categories of temporal logics

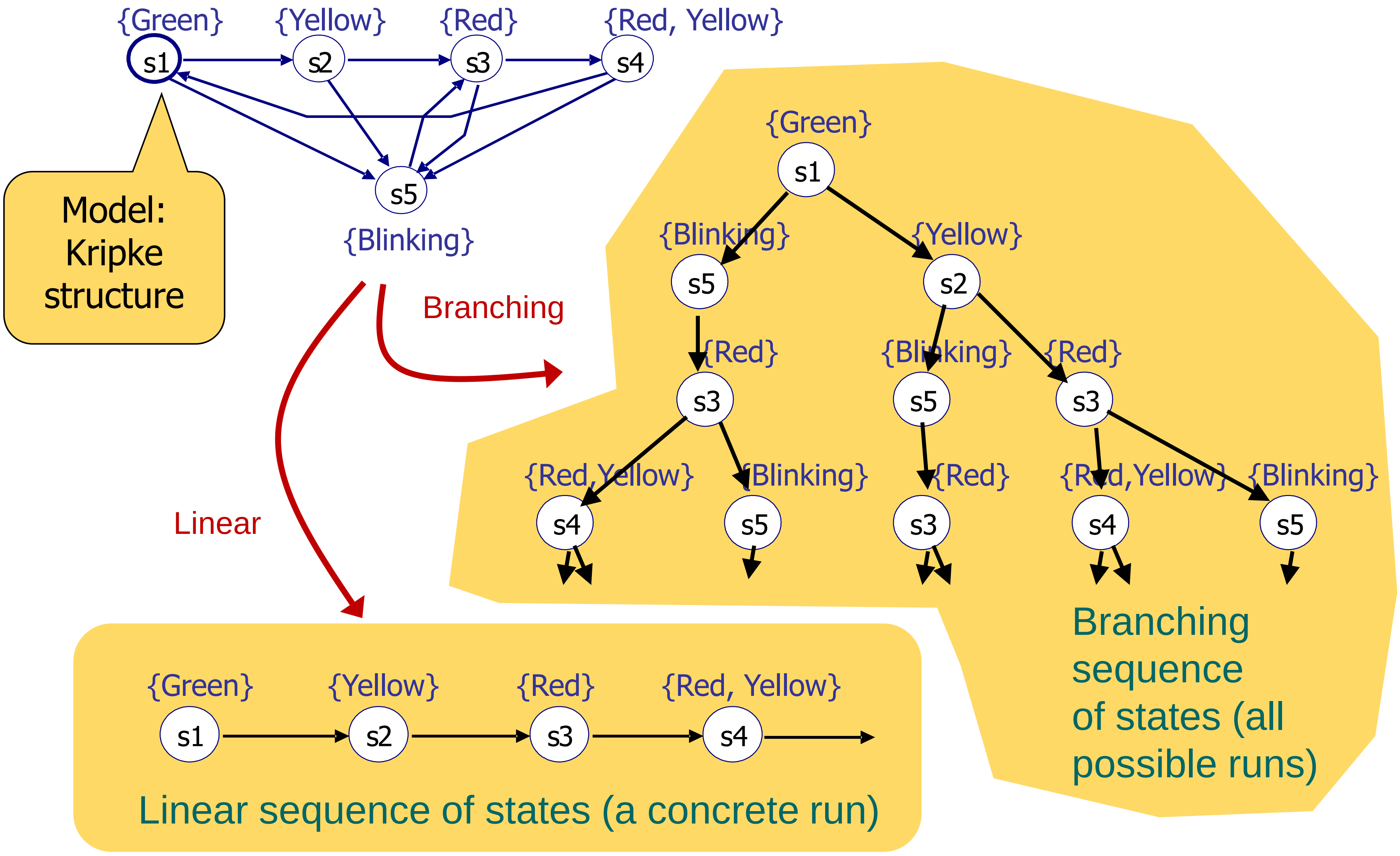
Formalization of reachability properties





- Goal: Formalizing state reachability properties
  - Occurrence of a state with local properties
    - Name, valuation of variables, mode of operation, …
  - Ordering of states: logical time
    - “Current” point in time: active state
    - “Subsequent” points in time: next state(s)
  - Typical reachability properties
    - Safety properties: Absence of “bad” state (universal property, invariant)
    - Liveness properties: Eventual occurrence of “good” state (existential property)
- Language used for formalization: Temporal logics
  - Formal system for evaluating changes in logical time
  - Temporal operators: “always”, “eventually”, “before”, “while”, …

Checking reachability properties



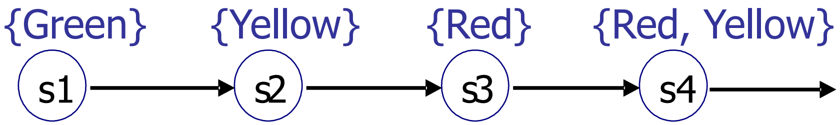
Model: Kripke structure

Branching sequence of states (all possible runs)

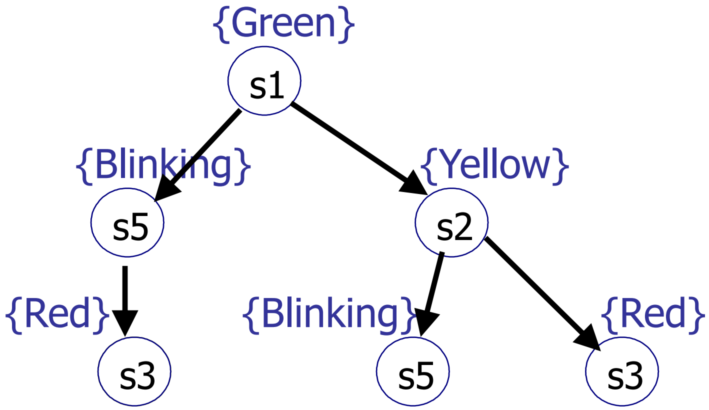
Linear sequence of states (a concrete run)

Categories of temporal logics

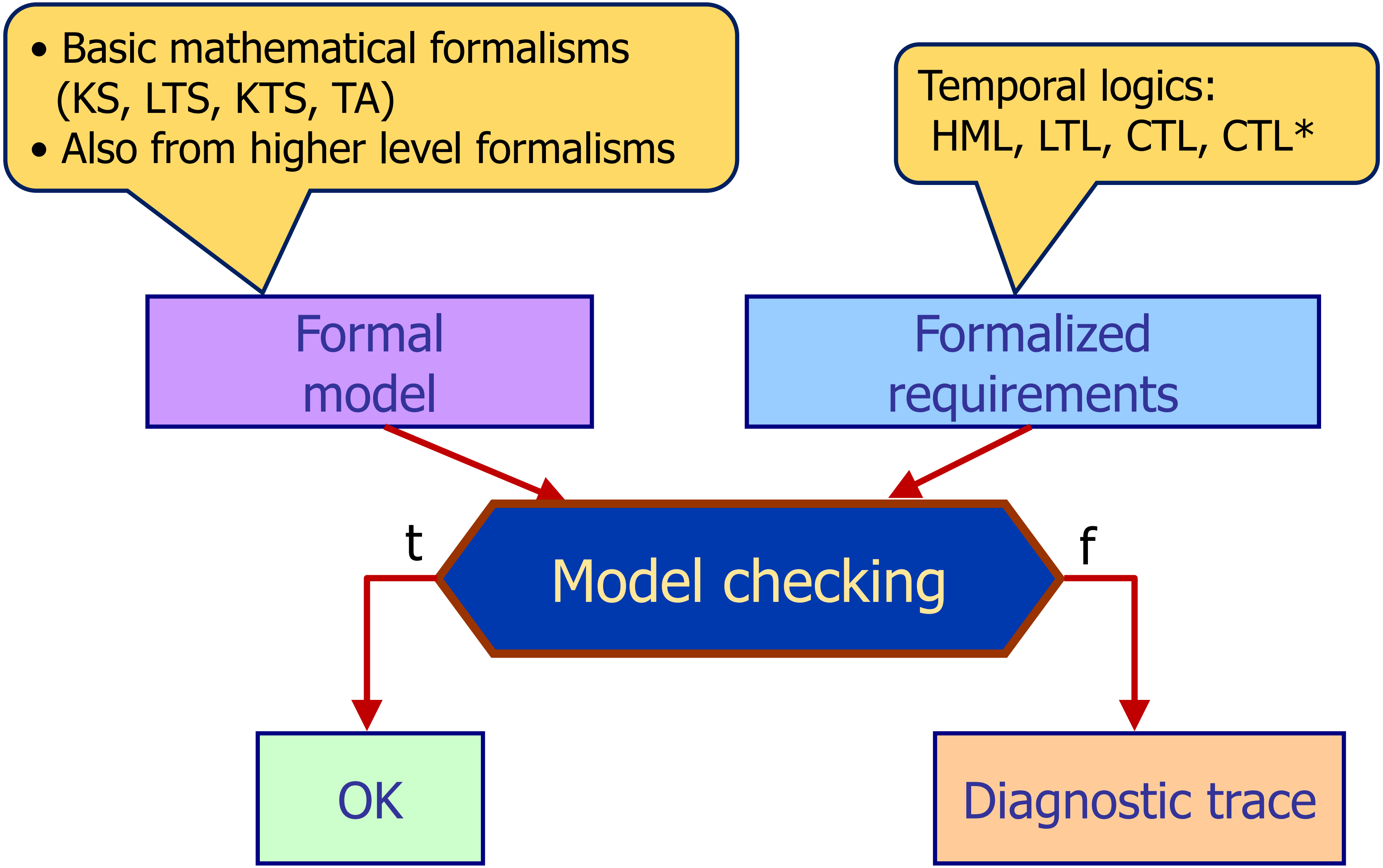
- Linear:
  - We consider individual executions of the system
  - Each state has exactly one subsequent state
  - Logical time along a linear timeline (trace)



- Branching:
  - We consider tree of executions of the system
  - Each state possibly has many subsequent state
  - Logical time along a branching timeline (so-called computation tree)



Model based verification by model checking



- Basic mathematical formalisms (KS, LTS, KTS, TA)
- Also from higher level formalisms

Temporal logics:
HML, LTL, CTL, CTL*

Formal model

Formalized requirements

Model checking

t

f

OK

Diagnostic trace

The Hennessy-Milner Logic

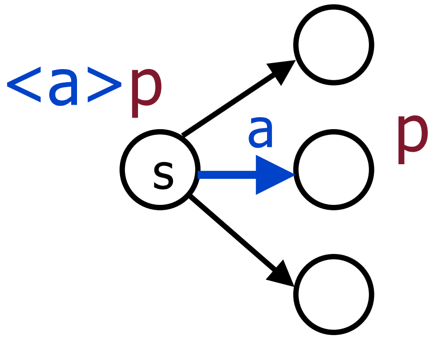
Syntax and semantics of temporal operators

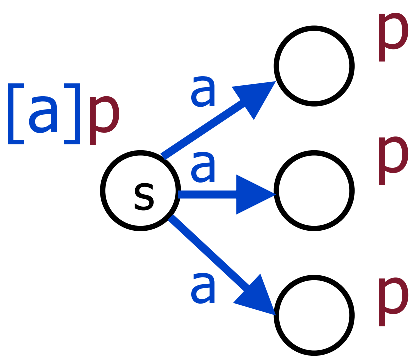
The Hennessy-Milner logic

- Simple logic interpreted on LTS $T=(S, Act, \rightarrow)$
- Properties of finite action sequences (scenarios, traces, test sequences) can be captured by HML
- Syntax: Rules for composing well-formed HML formula (p and q are well-formed formula, a is an action):

$$\mathrm{HML} ::= \mathrm{true} \mid \mathrm{false} \mid p \land q \mid p \lor q \mid [a]p \mid \langle a \rangle p$$

- Informal semantics of temporal operators:





Formal semantics of the Hennessy-Milner logic

Model of HML: LTS $T=(S, Act, \rightarrow)$

Semantic rules: Specify when p is true (satisfied) on state s of LTS T

Notation: $T,s \models p$

- $T,s \models true$, $T,s \not\models false$
- $T,s \models p \wedge q$ if and only if (iff) $T,s \models p$ and $T,s \models q$
  $T,s \models p \vee q$ iff $T,s \models p$ or $T,s \models q$
- $T,s \models [a]p$ iff $\forall s'$ where $s \rightarrow^a s'$: $T,s' \models p$
- $T,s \models \langle a \rangle p$ iff $\exists s'$: $s \rightarrow^a s'$ and $T,s' \models p$

HML examples:

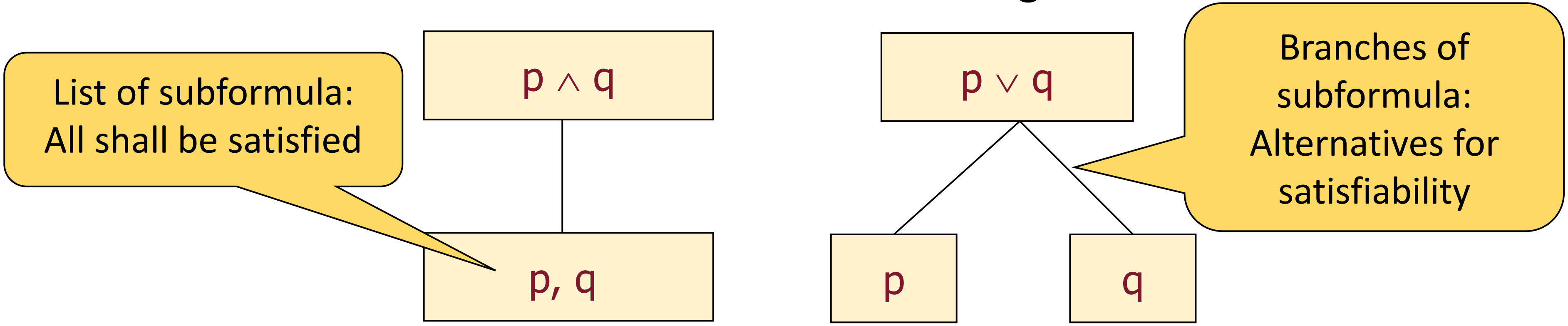
- <a>true: satisfied if there exists an outgoing transition labeled with a
- [a]false: satisfied if there is no outgoing transition labeled with a
- <a><b><c>true: satisfied if there exists an action sequence a,b,c

Checking Hennessy-Milner Logic properties

Tableau-based method

Introduction: Tableau for Boolean logic

- Goal: Determine the satisfiability of a logic formula
  - Decomposing the original formula into subformulas in a tree structure
  - In each node: subformula of the original formula to be satisfied
  - Branches: determined by construction rules
- Construction rules of the tableau for Boolean logic


List of subformula: All shall be satisfied
p ∧ q
p, q
p ∨ q
p
q
Branches of subformula: Alternatives for satisfiability

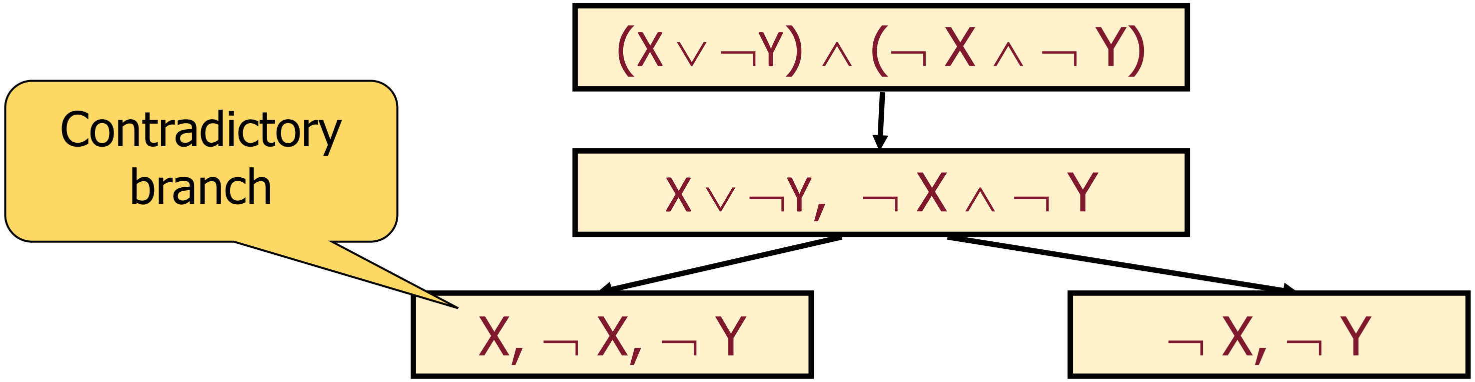
- Before decomposition, the formula shall be transformed to a negated normal form:
  - Negation only before variables (literals) and not before a composite formula
  - De Morgan’s laws can be used

Evaluation of the tableau

- Termination (closing) of a decomposition branch
  - Only a list of variables or negated variables is found in the current node
  - The satisfaction of the formula is given by assigning true to normal and false to negated variables
    - E.g., in case of p, ¬q the assignment is: p is true, q is false
- After the termination of a decomposition branch:
  - "Contradictory" branch: A variable is found both with and without negation (i.e., there is no valid assignment)
    - E.g., contradiction in case of p, ¬p, q
  - "Successful" branch: There is no contradiction, the valid assignment satisfies the original formula
- The "successful" branches determine the satisfiability of the original formula

Example: Tableau for a Boolean logic formula

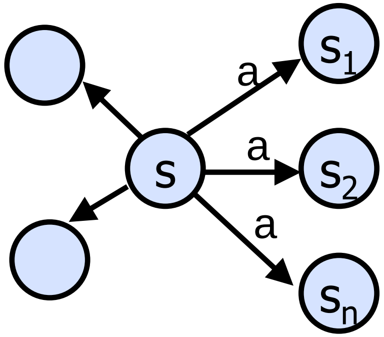
- Original formula: $\neg ((X \vee \neg Y) \rightarrow (X \vee Y))$
- Implication resolved: $\neg (\neg(X \vee \neg Y) \vee (X \vee Y))$
- Negated normal form: $(X \vee \neg Y) \wedge \neg(X \vee Y)$
  $(X \vee \neg Y) \wedge (\neg X \wedge \neg Y)$
- Construction of the tableau:


$(X \vee \neg Y) \wedge (\neg X \wedge \neg Y)$
$X \vee \neg Y, \ \neg X \wedge \neg Y$
$X, \neg X, \neg Y$
$\neg X, \neg Y$
Contradictory branch

- One of the branches is contradictory, the other is not
  - The original formula can be satisfied

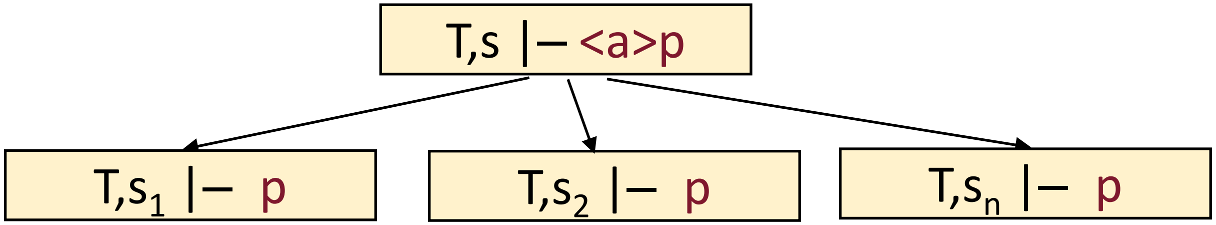
Checking HML by tableau construction (1)

- Tableau construction
  - Boolean operators: Branches as in case of Boolean tableau construction
  - Temporal operators:
    - Shall be evaluated on the outgoing transitions of a state s of the LTS T
    - The state of the LTS is part of the tableau
    - Moving to a next state if a transition is considered to satisfy a property
    - Notation: T,s |– p means looking for the satisfiability of property p in s of T
- Construction rules for the HML temporal operators:



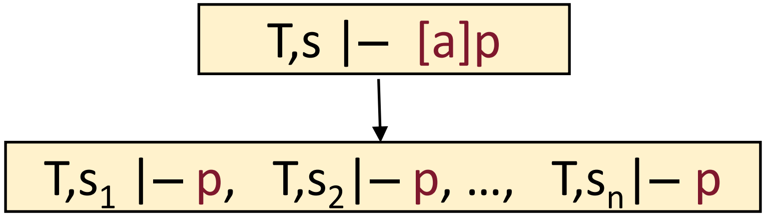
where

$\{s_1,s_2,..,s_n\}=\{s' \mid s \rightarrow^a s'\}$



$T,s\ |\!-\ \langle a \rangle p$

$T,s_1\ |\!-\ p$ $\quad$ $T,s_2\ |\!-\ p$ $\quad$ $T,s_n\ |\!-\ p$



$T,s\ |\!-\ [a]p$

$T,s_1\ |\!-\ p,\quad T,s_2\ |\!-\ p,\ \dots,\quad T,s_n\ |\!-\ p$

Checking HML by tableau construction (2)

- The construction of the tableau is based on the model
- Successful branches:
  - If T,s |- true is reached
  - If T,s |- [a]p is reached where there is no outgoing transition labeled with a
- The role of successful branches:
  - If the original formula is negated before the construction of the tableau:
    The successful branch provides a counter-example for the original property
  - This is the approach used in model checking

Summary

- Formalization of requirements
  - Frequent patterns
- Categories of temporal logics
  - Linear time temporal logics
  - Branching time temporal logics
- HML: Hennessy-Milner logic
  - Temporal operators
  - Model checking: Tableau method