

Propositional Scopes in Linear Temporal Logic

M. Haydar, S. Boroday,
A. Petrenko, and H. Sahraoui

NOTERE'05 - August 31, 2005



Outline

- Motivation
- Problem description
- Propositional scopes in LTL
- System of property patterns
- Conclusion

Motivation

- Model checking is increasingly used for the analysis of distributed systems
 - Automation
 - Preciseness
- The price is high in terms of complexity
 - State explosion
 - Current programming languages are quite versatile
 - Learning some property languages is not easy

Motivation

- Property specification is difficult for novices and professionals
 - Lack of complete specifications
 - Limitations of the languages
- Properties can become very complex when only part of the system is considered

Model Checking

- Check if a system M satisfies a property φ

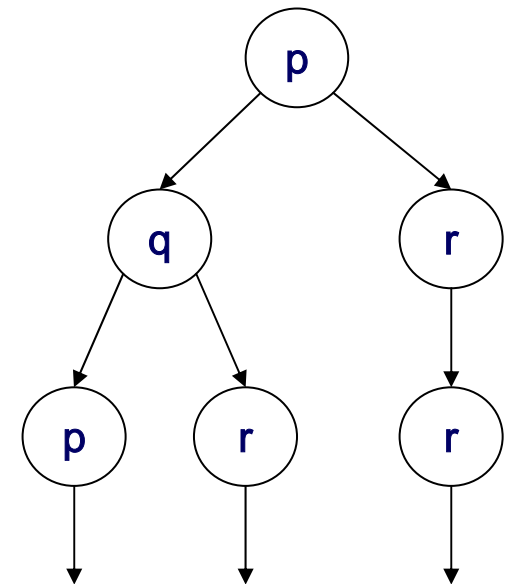
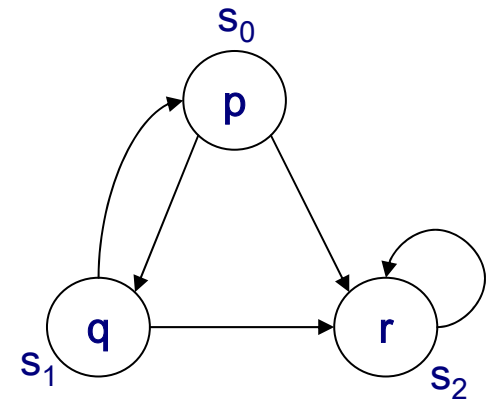
- System

Kripke structure (S, S_0, R, L)

- S : finite set of states
- $S_0 \subseteq S$: set of initial states
- $R \subseteq S \times S$: transition relation
- $L: S \rightarrow 2^{AP}$ set of atomic propositions true in each state

- Property

Temporal formula or automaton



Overview of LTL

■ Syntax

$$\varphi ::= p \mid (\neg\varphi) \mid (\varphi \wedge \varphi) \mid (\varphi \cup \varphi) \mid (G \varphi) \mid (F \varphi) \mid (X \varphi)$$

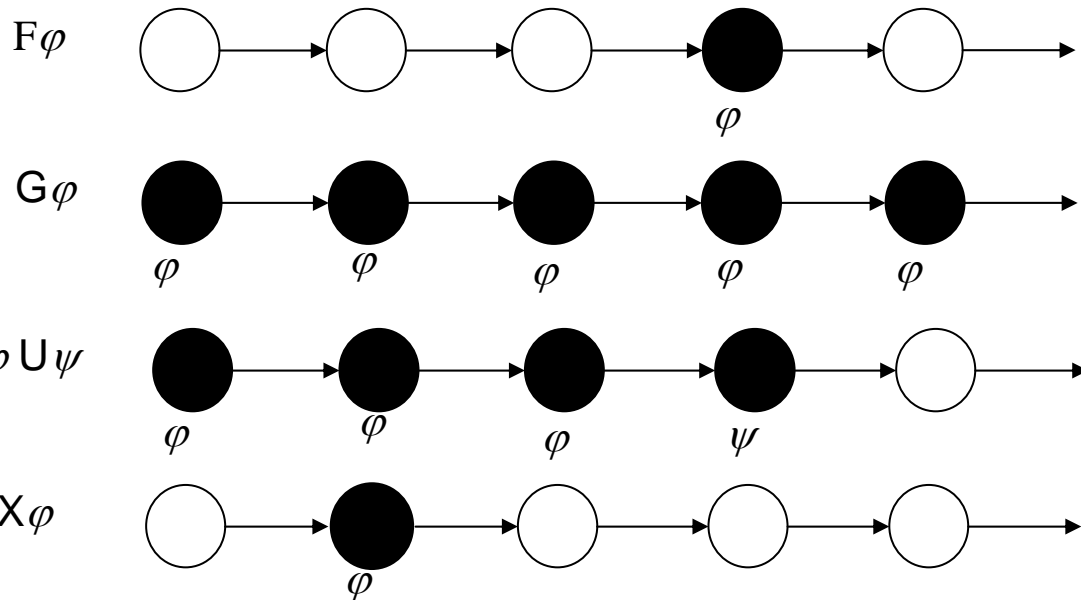
■ Semantics

F: eventually

G: always

U: until

X: next



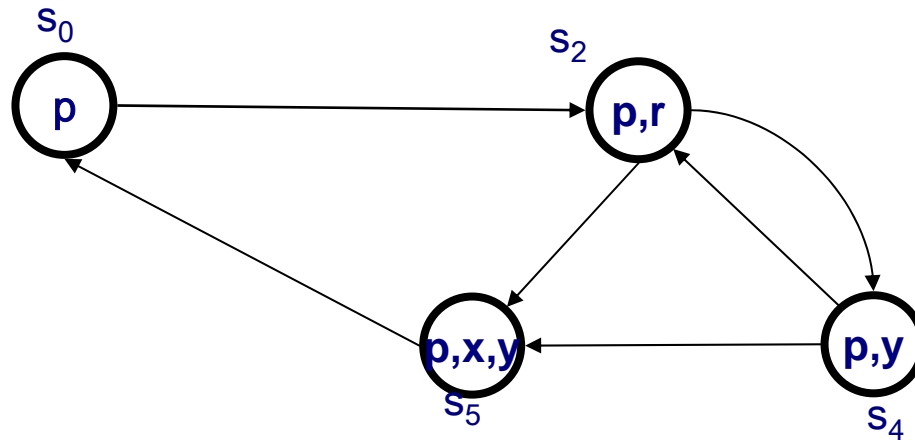
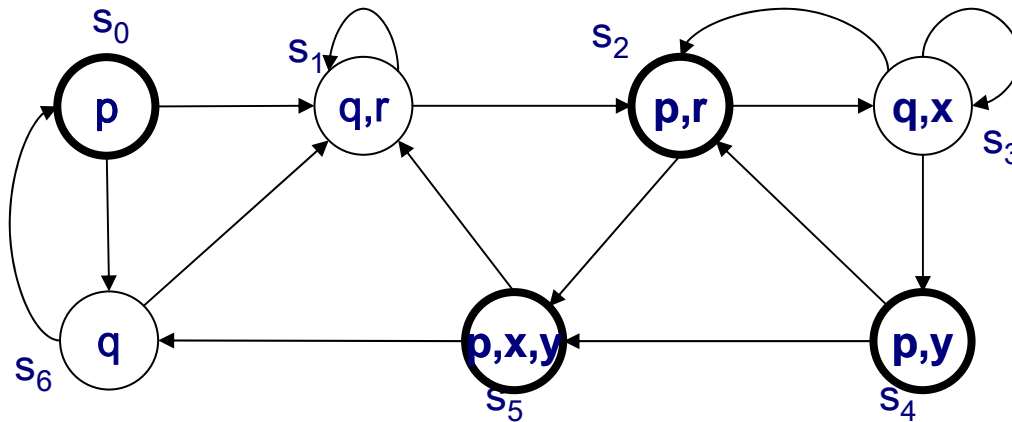
LTL is not Enough

- Expressiveness of LTL is limited
- System model may include states uninteresting to certain properties
- How to specify LTL properties checkable in a subset of the state space

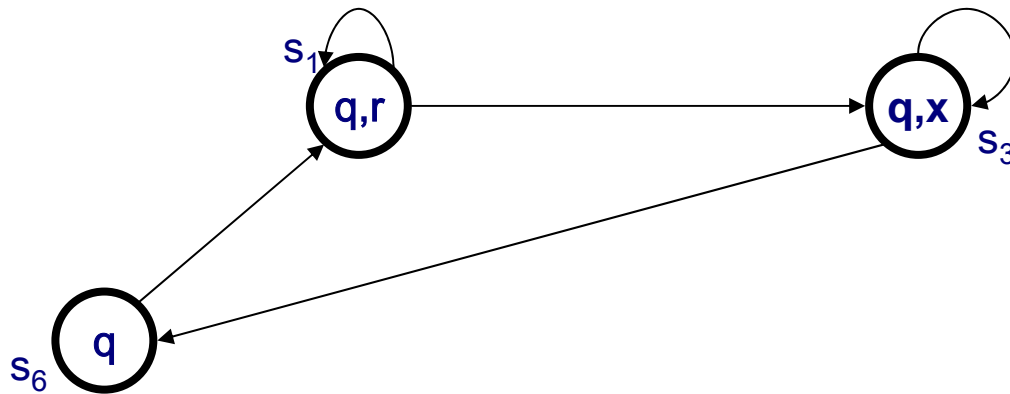
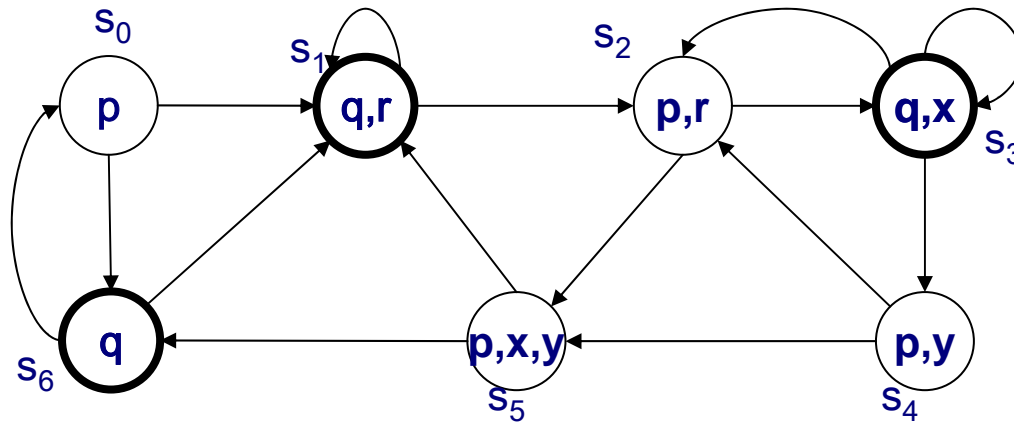
Intuitive solution

remove the uninteresting states from the model

Projection of the Model

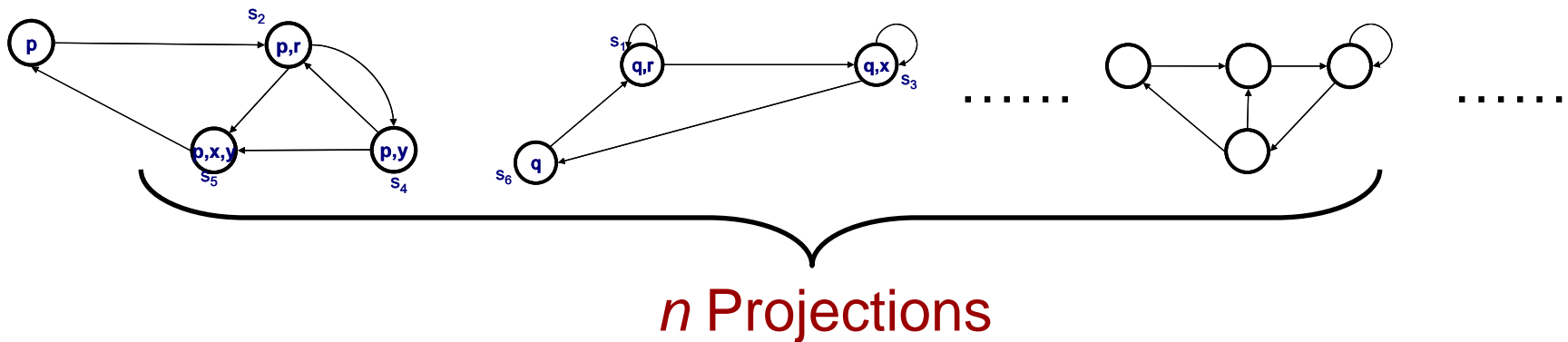
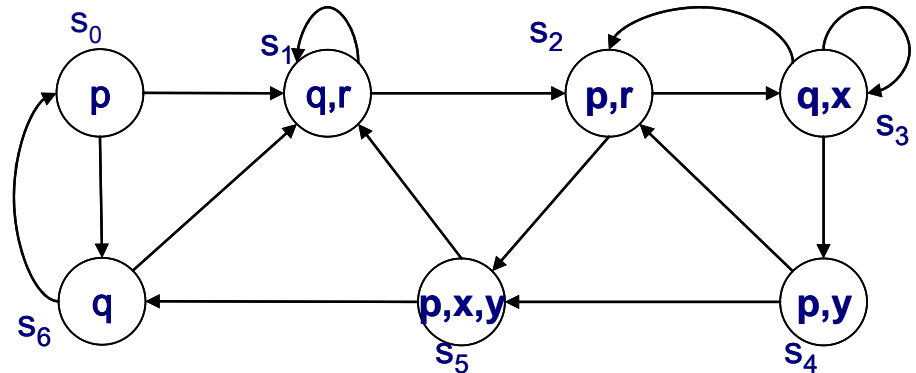


Projection of the Model



Projection of the Model

- Unfeasible solution
- Worst case
 - n properties
 - n subsets of states



- Systems are usually modular
- Model Checkers use higher level languages

Problem Statement

- Define a means to specify LTL properties over arbitrary subsets of the state space
 - Do not add to the complexity of specification
 - Do not modify the model
 - Model checking algorithms still can be used

Solution

- Introduce scopes to define subsets of the search state space
 - Defined by propositional expressions
 - Help determine arbitrary subsets of the state space
- Add new operators to LTL to check properties within defined scopes
- Result in succinct LTL formulae

LTL with Scopes

- Eventually in scope

$$F_{\mathcal{I}} \varphi = F (\varphi \wedge \mathcal{I})$$

- Always in scope

$$G_{\mathcal{I}} \varphi = G (\mathcal{I} \rightarrow \varphi)$$

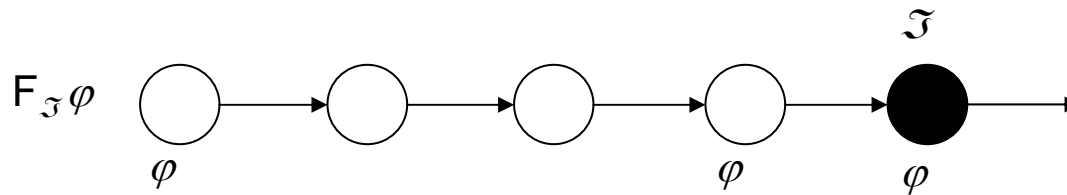
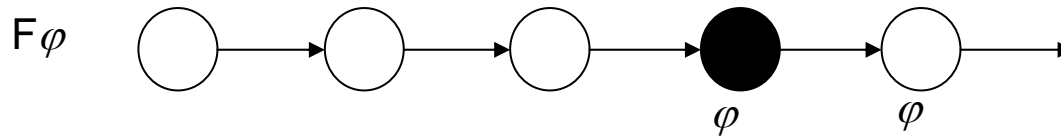
- Until in scope

$$\varphi U_{\mathcal{I}} \psi = (\mathcal{I} \rightarrow \varphi) U (\psi \wedge \mathcal{I})$$

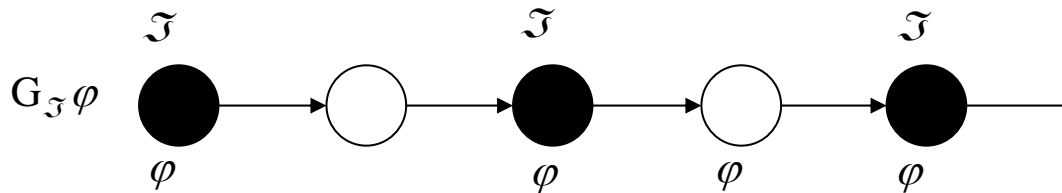
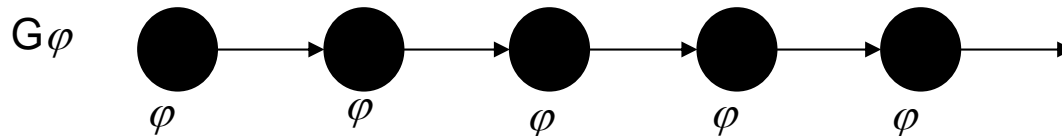
- Next in scope

$$X_{\mathcal{I}} \varphi = \neg \mathcal{I} U [\mathcal{I} \wedge X (\neg \mathcal{I} U (\mathcal{I} \wedge \varphi))]$$

How It Works

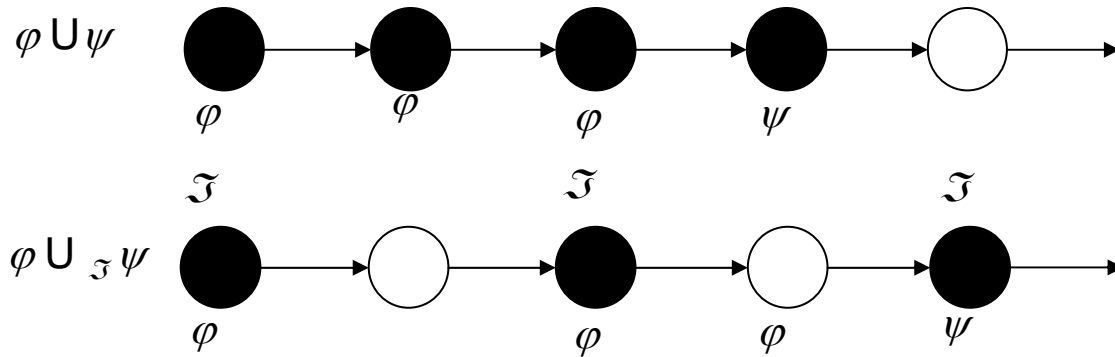


$$F_{\mathcal{J}}\varphi = F(\varphi \wedge \mathcal{J})$$

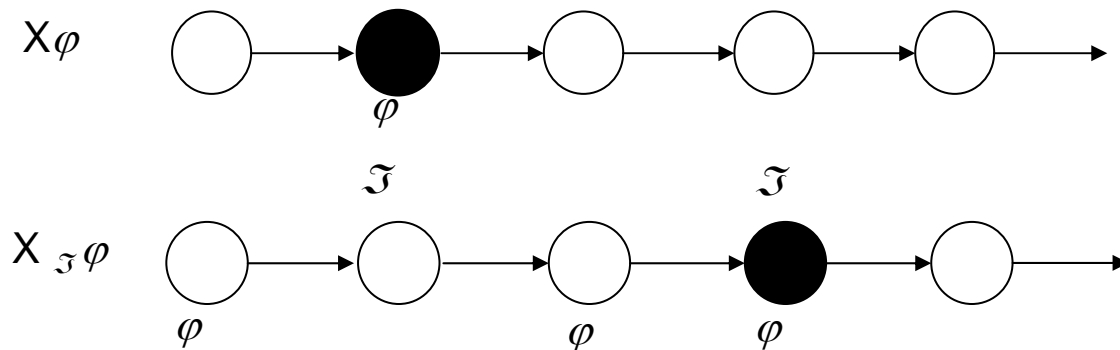


$$G_{\mathcal{J}}\varphi = G(\mathcal{J} \rightarrow \varphi)$$

How It Works



$$\varphi U_{\mathcal{T}} \psi = (\mathcal{T} \rightarrow \varphi) U (\psi \wedge \mathcal{T})$$



$$X_{\mathcal{T}} \varphi = \neg \mathcal{T} U [\mathcal{T} \wedge X (\neg \mathcal{T} U (\mathcal{T} \wedge \varphi))]$$

Scope operators

Operator **In**

- Recursively applies propositional scope on any formula
- Makes property specification intuitive and succinct

$$p \text{ In } \mathcal{I} = \neg \mathcal{I} \cup (p \wedge \mathcal{I})$$

$$(\neg \varphi) \text{ In } \mathcal{I} = \neg \mathcal{I} (\varphi \text{ In } \mathcal{I})$$

$$(\varphi \wedge \psi) \text{ In } \mathcal{I} = (\varphi \text{ In } \mathcal{I}) \wedge_{\mathcal{I}} (\psi \text{ In } \mathcal{I})$$

$$(\varphi \cup \psi) \text{ In } \mathcal{I} = (\varphi \text{ In } \mathcal{I}) \text{ In }_{\mathcal{I}} (\psi \text{ In } \mathcal{I})$$

$$(G \varphi) \text{ In } \mathcal{I} = G_{\mathcal{I}} (\varphi \text{ In } \mathcal{I})$$

$$(F \varphi) \text{ In } \mathcal{I} = F_{\mathcal{I}} (\varphi \text{ In } \mathcal{I})$$

$$(X \varphi) \text{ In } \mathcal{I} = X_{\mathcal{I}} (\varphi \text{ In } \mathcal{I})$$

Correctness

- **Theorem 1.** *For any LTL formula φ and its corresponding formula φ **In** \mathfrak{S} , $\pi \models \varphi$ **In** \mathfrak{S} iff $\pi_{\downarrow \mathfrak{S}} \models \varphi$.*
- **Theorem 2.** *Let \mathfrak{S} be a propositional logic formula, and let $M = (S, T, S_0, L)$ and $M_{\mathfrak{S}} = (S_{\mathfrak{S}}, T_{\mathfrak{S}}, S_{0\mathfrak{S}}, L_{\mathfrak{S}})$ be two Kripke structures, $M_{\mathfrak{S}}$ is the projection of M onto $S_{\mathfrak{S}} \subseteq S$ such that \mathfrak{S} is true in all $s \in S_{\mathfrak{S}}$. For any LTL formula φ , $M \models \varphi$ **In** \mathfrak{S} iff $M_{\mathfrak{S}} \models \varphi$.*

Example

- Property (DWR)

*Robot visits **Room1** then **Room2**, then visits **Room1**, and repeat this exactly **twice***

- Standard LTL

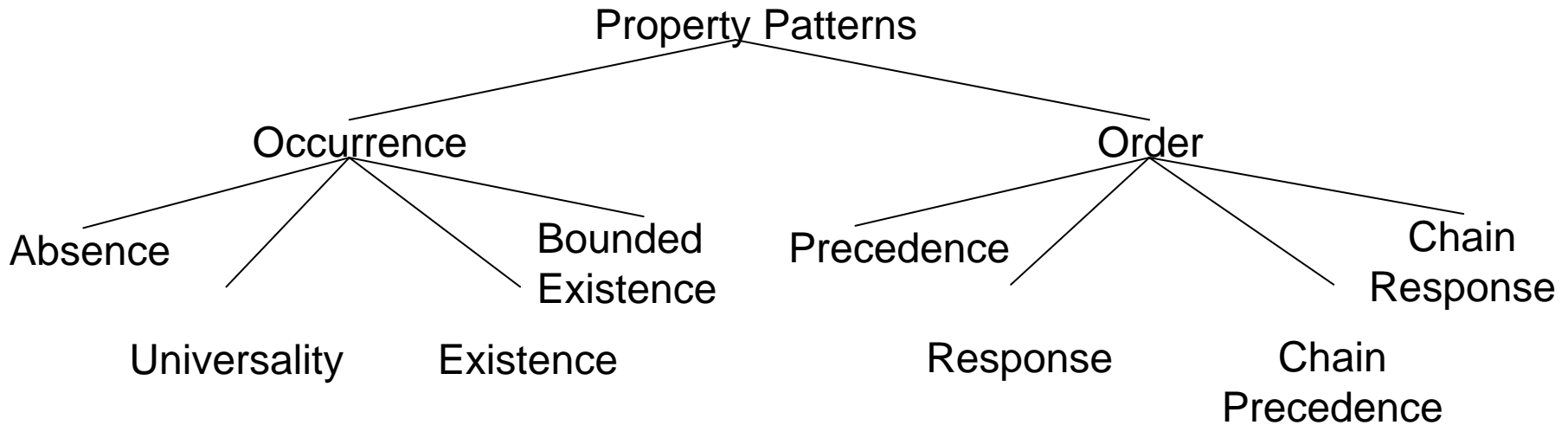
$$\begin{aligned} & (\neg \text{Room1} \wedge \neg \text{Room2}) \text{ U } (\text{Room1} \wedge \neg \text{Room2} \wedge \text{X} \\ & (\neg \text{Room2} \text{ U } (\text{Room2} \wedge \neg \text{Room1} \wedge \text{X} (\neg \text{Room1} \text{ U } \\ & (\text{Room1} \wedge \neg \text{Room2} \wedge \text{X} (\neg \text{Room2} \text{ U } (\text{Room2} \wedge \\ & \neg \text{Room1} \wedge \text{X} (\neg \text{Room1} \text{ U } (\text{Room1} \wedge \neg \text{Room2} \wedge \\ & \text{X} (\text{G} (\neg \text{Room2})))))))))) \end{aligned}$$

Example

- Using **In** operator

- Scope is $(Room1 + Room2)$ where $+$ is the exclusive or operator
- $Room1 \wedge X(Room2 \wedge X(Room1 \wedge X(Room2 \wedge X(Room1 \wedge G(\neg Room2)))))) \mathbf{In} (Room1 + Room2)$

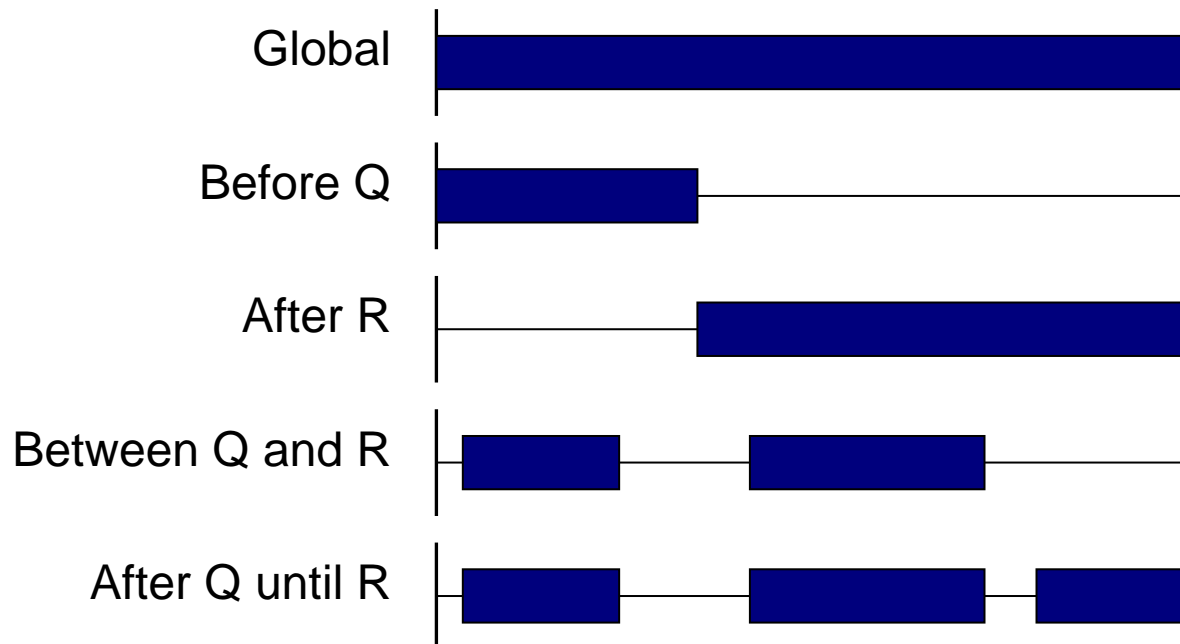
System of Property Patterns



- **Absence** - A state/event does not occur within a scope
- **Existence** - A state/event must occur within a scope
- **Universality** - A state/event occurs throughout a scope
- **Response** - A state/event P must be always followed by a s/e Q within a scope
- **Precedence** - A state/event P must be always preceded by a state/event Q within a scope

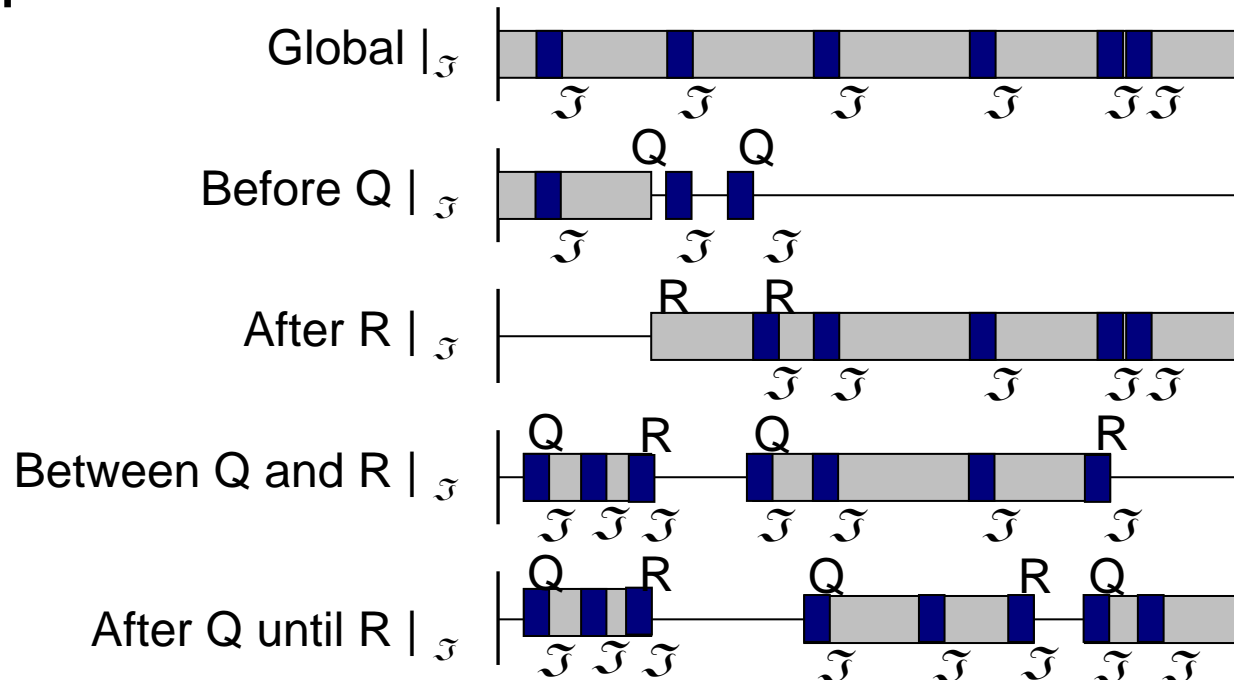
System of Property Patterns

- Five scopes are defined
- A scope is an interval of states that has a start and end state



Combination of Scopes

- We combined our scope with Dwyer's scopes
- Enrich the expressiveness of patterns
- More flexibility for specification of real world properties



Conclusion

- We introduced new LTL scope operators
 - Specify properties over arbitrary subsets of states identified by propositional expressions
 - Help in more succinct and intuitive specification
- We combined our scope with Dwyer's scopes
 - 5 additional scopes are defined
- We applied our scope operators on property patterns

Future Work

- Extend our approach to temporal scopes
 - Helps in the improvement and elucidation of property patterns
 - Provides richer framework for automated recursive application of temporal scopes on arbitrary formula
- Implement the translation of a scoped LTL formula into standard LTL



Thank you!