Model Checking of Location and Mobility Related Security Policy Specifications in Ambient Calculus

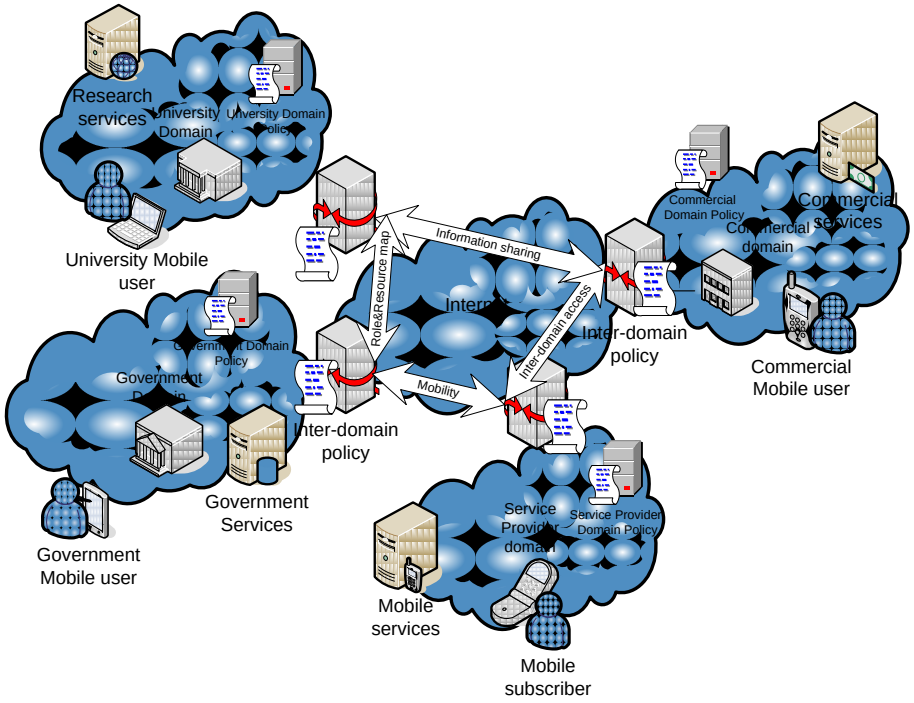
Devrim Ünal (presenter)
devrimu@uekae.tubitak.gov.tr
National Institute of Electronics and Cryptology, TUBITAK, Turkey

Ozan Akar, Prof. Dr. M. Ufuk Caglayan
Computer Networks Research Laboratory (NETLAB), Bogazici University, Turkey

Outline of Presentation

- Introduction
- Motivation
- Related Work
- A Formal Model for Security Policies for Multi-domain Mobile Networks in Ambient Calculus
- Model Checking of Security Policy Specifications in Ambient Calculus Model Checker
- Complexity and Performance Analysis
- Conclusion

Introduction (Problem Domain)



- The *multi-domain mobile network environment* consists of multiple interconnected domains and mobile users, hosts and objects.
- Interconnection and mobility are the two main concepts that come into consideration where users are allowed to use network connectivity of multiple domains.

Introduction (Contributions)

- We present a method and a model checking tool for formal specification and verification of location and mobility related security policies for mobile networks.
- The formal languages used for specification are Predicate Logic and Ambient Calculus.
- The presented tool is capable of spatial model checking of Ambient Calculus specifications for security policy rules and uses the NuSMV model checker for temporal model checking.

Motivation

- Security policies in a multi-domain environment require specification and verification of multiple policies.
- The heterogeneous nature of mobile networks suggests a common formal policy model.
- Security breaches in multi-domain mobile networks often arise from insufficient representation and enforcement of multiple actions including mobility.
- Formal verification of policies by use of a common formal policy model provides means to reduce security breaches arising from incomplete, inconsistent and ambiguous specification of multi-domain policies.

Related Work (1)

- Formal security policy models
  - SECPAL (Becker et al.)
  - Flexible Authorization Framework (Jajodia et al.)
  - Deontic Logic (Cuppens et al.)
  - Various RBAC Models (GT-RBAC, GEO-RBAC, STARBAC, Lot-RBAC,...)
- Verification of policies
  - Theorem proving for RBAC policies (Sohr et al.)
  - ACPEG – First order logic (Zhang et al.)
  - Theorem proving with Coq – (Unal et al.)
  - MARGRAVE (?)

Related Work (2)

- Security Policy Specifications in Ambient Calculus
  - Application-level security policies for ubiquitous computing (Scott)
  - $BACI_R$: Role-based access control for Ambient Calculus (Compagnoni et al.)
- Model Checking with Ambient Calculus
  - Model checking biological systems (Mardare et al.)
  - Model checking mobile ambients (Charatonik et al.)

A Formal Model for Security Policies for Multi-domain Mobile Networks in Ambient Calculus

- Formal Model for Security Policy
- Formal Specification of Mobile Processes in Ambient Calculus
- Formalization of Location and Mobility Related Policy Constraints

Formal Model for Security Policy

- Data Sets and Relations
- Authorization Terms
- Role Based Access Control Model
- Location and Mobility Constraints

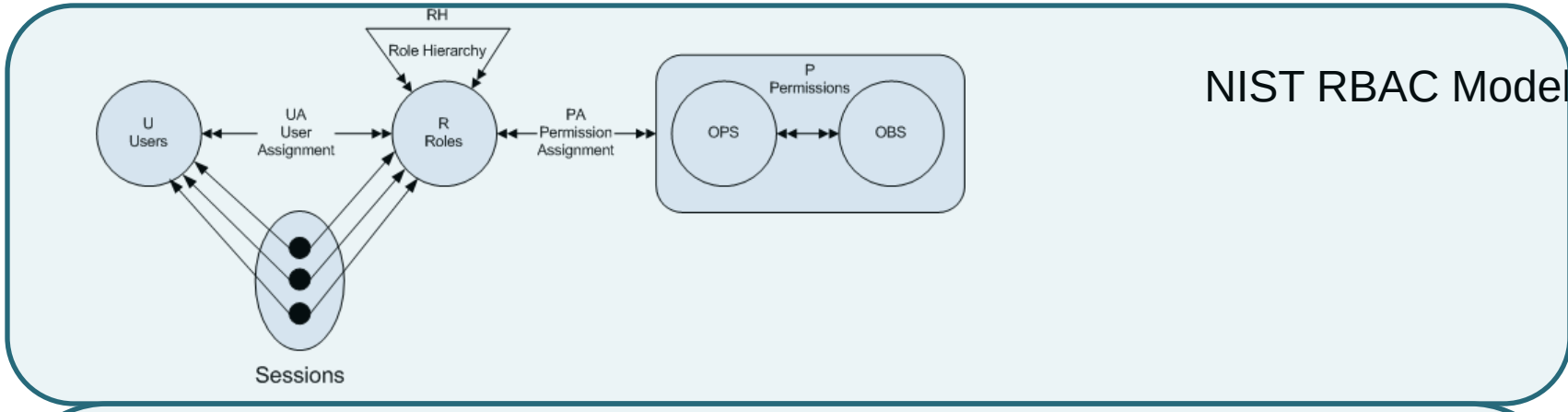
Data Sets and Relations

$D = \{D_1, D_2, \ldots D_d\}$	Domains
$H = \{H_1, H_2, \ldots H_n\}$	Hosts
$U = \{U_1, U_2, \ldots U_m\}$	Users
$R = \{R_1, R_2, \ldots R_o\}$	Roles
$O = \{O_1, O_2, \ldots, O_t\}$	Objects
$OT = \{OT_1, OT_2, \ldots OT_v\}$	Object types
$AS = R \cup U$	Authorization Subjects
$AO = O \cup OT \cup H \cup D$	Authorization Objects
$HOD: H \times D$	Relation mapping hosts to domains.
$UOD: U \times D$	Relation between users and domains.
$OOT: O \rightarrow OT$	Function that specifies the type of an object.
$UA: U \times R$	Relation for assignment of users to roles.
$PA: R \times SA \times AO$	Relation for assignment of roles to permissions.

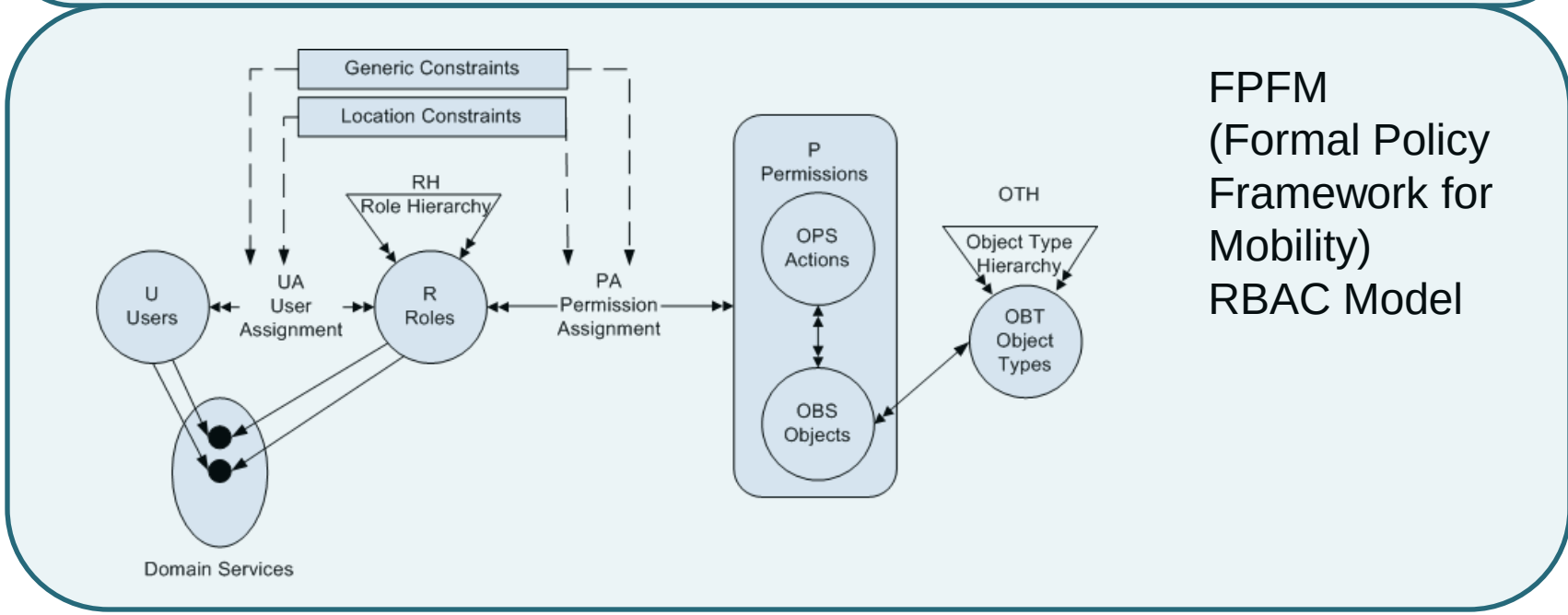
Authorization Terms

- The basic security policy rule is the authorization term *at = (as, ao, sa, co, fo)*
  - *as*: Authorization Subject (Roles)
  - *ao*: Authorization Object (Domains, Hosts, Objects, Object Types)
  - $sa \in S \times A$ is a signed action of the form (+,read)
  - *co*: Condition (a generic constraint) is a first-order logic formula from pre-defined predicates,
  - *fo*: Formula is an *Ambient Logic* formula that defines location and mobility constraints.
- Security Policy SP is a set of authorization terms.

Role Based Access Control Model



NIST RBAC Model



FPFM
(Formal Policy Framework for Mobility)
RBAC Model

Location and Mobility Constraints

Constraint	Represents	Represented by
Location	Locations of objects, users, hosts and domains	Somewhere modality, parallel and ambient formalizations of ambient logic
Mobility	The change of locations with time	Sometime and Everytime modalities in ambient logic

Location and mobility constraints in the security policy rule.

- Location constraints are verified using Spatial Model Checking
- Mobility constraints are verified using Temporal Model Checki ng.

Formal Specification of Mobile Processes in Ambient Calculus

- Ambient Calculus

P, Q	::=	processes	M	::=	capabilities
	ε	inactivity		x	variable
	$P \mid Q$	composition		n	name
	$M[P]$	ambient		$in \quad M$	can enter M
	$M.P$	capability		$out \quad M$	can exit M
	$(x).P$	input		$open \quad M$	can open M
	$\langle M \rangle$	asynchronous output		0	null
				$M.M$	path

- Ambient Logic

$\mathcal{A}, \mathcal{B}, \mathcal{C}$	η	a name n		
::=	T	true	$\mathcal{A} \mid \mathcal{B}$	composition
	$\neg \mathcal{A}$	negation	$n[\mathcal{A}]$	location
	$\mathcal{A} \vee \mathcal{B}$	disjunction	$\Diamond \mathcal{A}$	sometime modality
	0	void	$\blacklozenge \mathcal{A}$	somewhere modality

Specification of Mobility in Ambient Calculus

- A message is sent from User 1 to User 3:

World [DomainA [Server1 [
User1 [message[M | out User1. out Server1. out DomainA. in DomainB, in Client2. in User3.0]]]] | DomainB [Client2 [User3 [open message.(m).0]]]] $\rightarrow$*

World [DomainA[Server1[User1[]]] |
DomainB [Client2[User3[M]]]]

- Portable 1 disconnects from Domain A and connects to Domain B:

World [DomainA[
Portable1[out DomainA. in DomainB .0]] | DomainB[]] $\rightarrow$*
World [DomainA[] | DomainB[Portable1[]]]

Formalization of Location and Mobility Related Policy Constraints

- ``Files in Host 2 can not be read from within Portable hosts.'':
  (*as* = User, *ao* = file, *sa* = (-) read,
  fo = (Portable [*as[]*] | Host2 [*ao*[]]), *co* = T)
- ``User 1 may not send messages to User 2.'':
  (*as* = User1, *ao* = User2, *sa* = (-) send,
  fo = *as* [message[] | T] *ao*[T], *co* = T)
- ``A user can connect Portable 1 to Domain B.'':
  (*as* = User, *ao* = DomainB, *sa* = (+) connect,
  fo = (Portable1 [*as[]*] | *ao* [T]), *co* = T)
- ``User 1 can login to Host 2 '':
  (*as* = User1, *ao* = Host2, *sa* = (+) login,
  fo = (*ao* [T] | *as* [T]), *co* = T)

Applying Location and Mobility Constraints within Policy Rules

- To check location constraints in security policy, the input to the model checker tool is an Ambient Calculus specification and a set of Ambient Logic formulas.
- To express that process P satisfies the formula *A* , $P \vdash A$ *is used.*
- The satisfaction relation $\vdash$ determines the security policy rules applicable in a given mobile network setting.

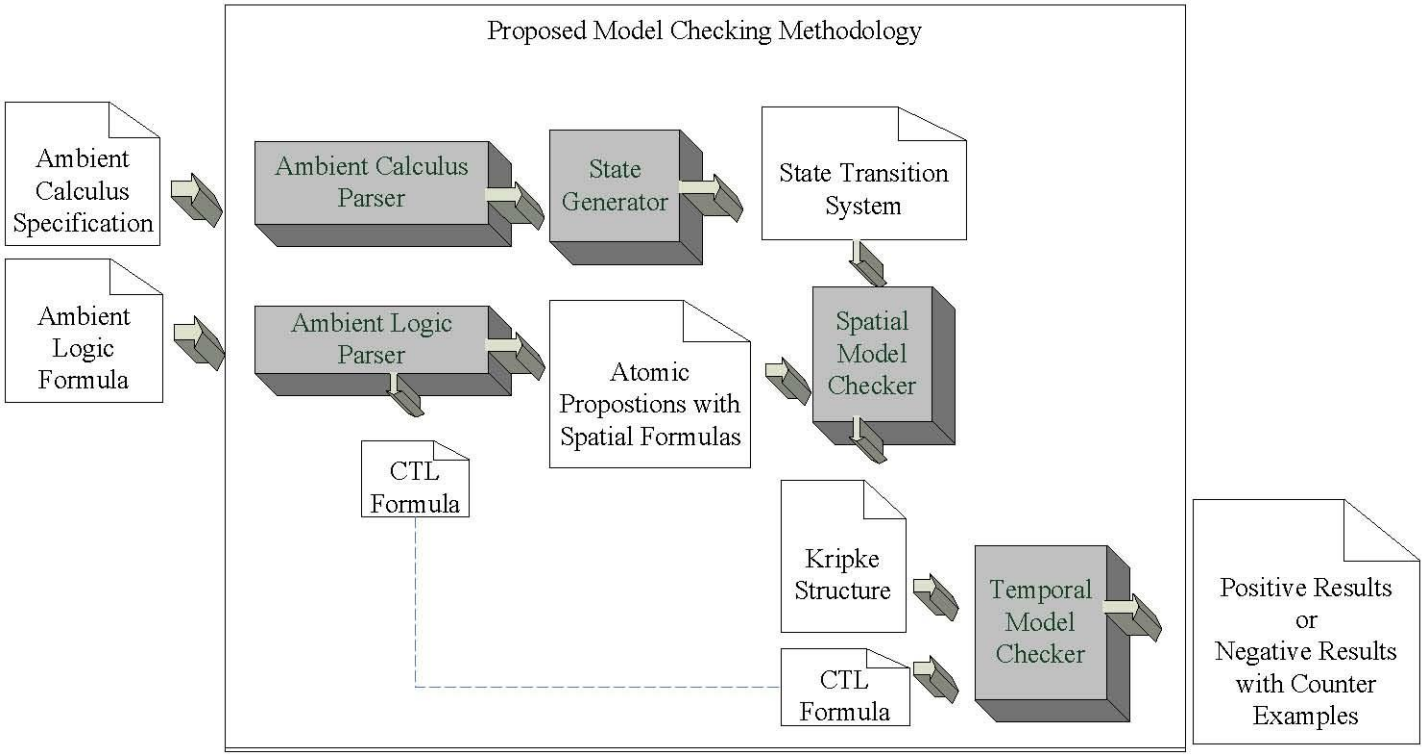
Model Checking of Security Policy Specifications in Ambient Calculus Model Checker

- Overview of Model Checking Algorithm
- Ambient Calculus Model Checker
- Ambient Topology and Spatial Formula Graphs
- State Transition System Generation
- Checking Spatial Modalities
- Generation of Kripke Structure
- NuSMV Code Generation
- Example for Spatial Model Checking Algorithm

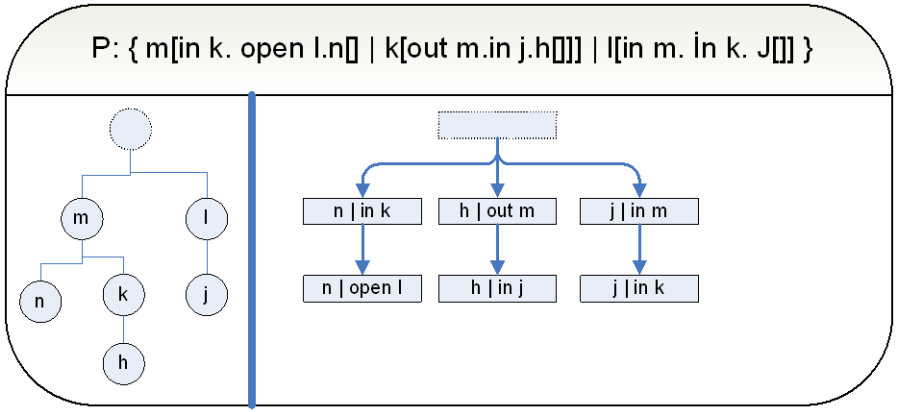
Overview of Model Checking Algorithm

1. Define atomic propositions with respect to spatial properties of ambient logic formula and register the (atomic proposition-spatial modality) couples.
2. Reduce ambient logic formula to temporal logic formula (CTL) by replacing spatial modalities with atomic propositions.
3. Generate state transition system of the ambient calculus specification with respect to reduction relations.
4. Generate Kripke Structure from state transition system.
5. Generate NuSMV code from Kripke Structure and CTL Formula.

Ambient Calculus Model Checker

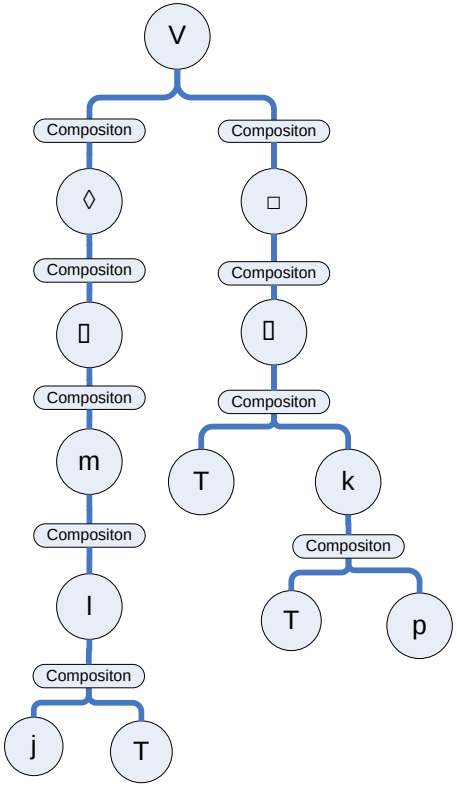


Ambient Topology and Capability Trees



- *Ambient Topology* is an acyclic digraph where elements of set of *nodes* denote ambients within the ambient calculus specification and arcs denote parent-child relation among ambients.
- *Capability Tree* is an acyclic digraph where set of nodes denote capabilities and arcs , denote priority relation among capabilities. Nodes contain the information about which ambient the capability is attached and which ambient the capability effects.

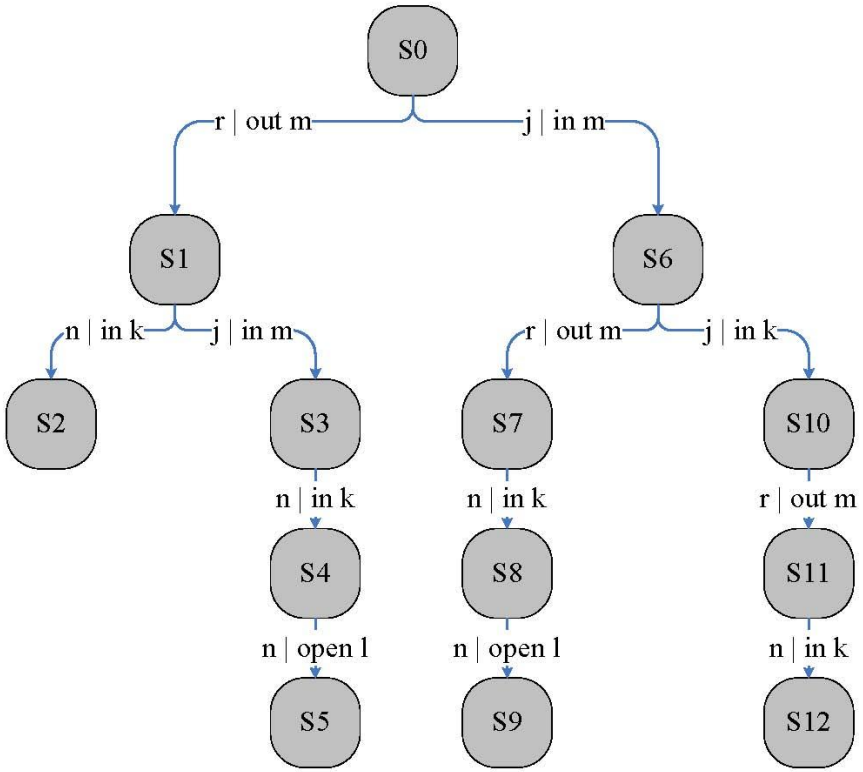
Spatial Formula Graphs



F: ◊(▯ (m [l [j[]]] | T)) V □(▯ (k[p[] | T] | T))

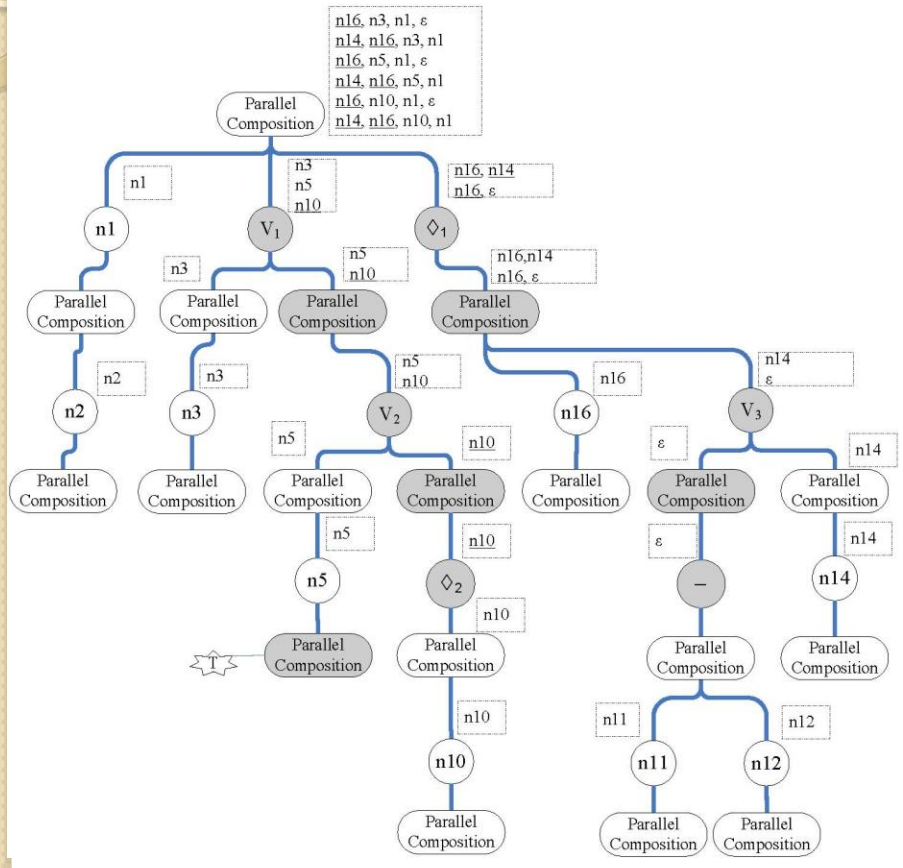
- A spatial formula graph is an acyclic digraph where node denotes connectives and locations, arcs denote the operator operand relation.

State Transition System Generation



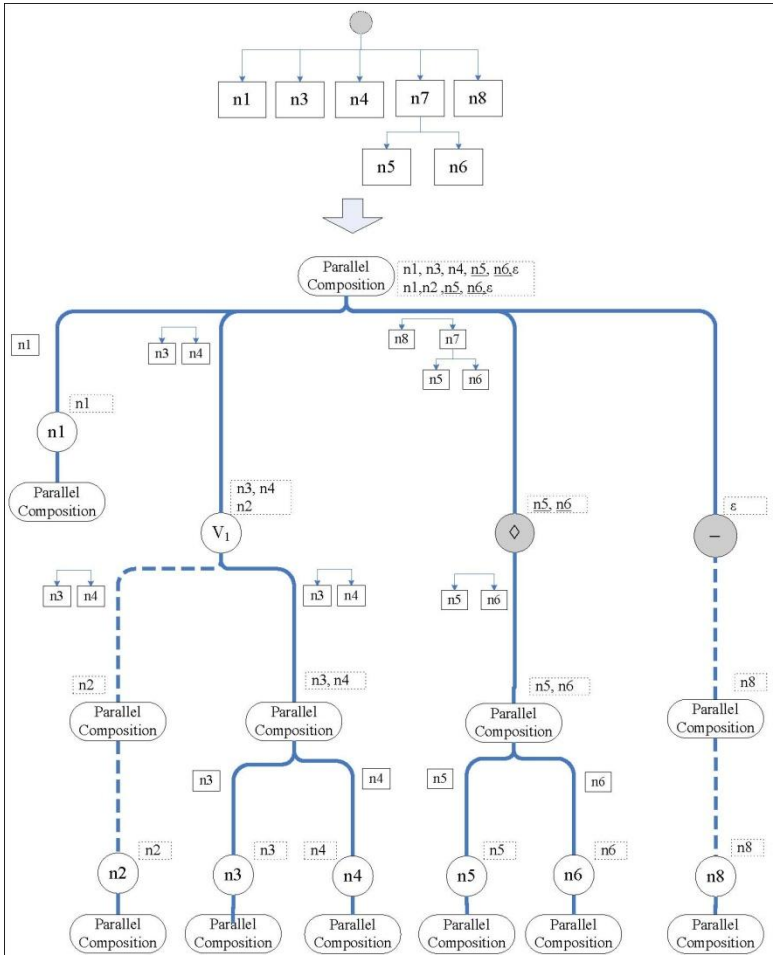
- State transition system can be represented by an acyclic digraph (replication-free)
- Nodes represent states and edges represent the execution of a capability
- For selection of the next capability to execute, some condition checks are carried out.
- These conditions are the location of the object ambient and the availability of the subject ambient.

Checking Spatial Modalities



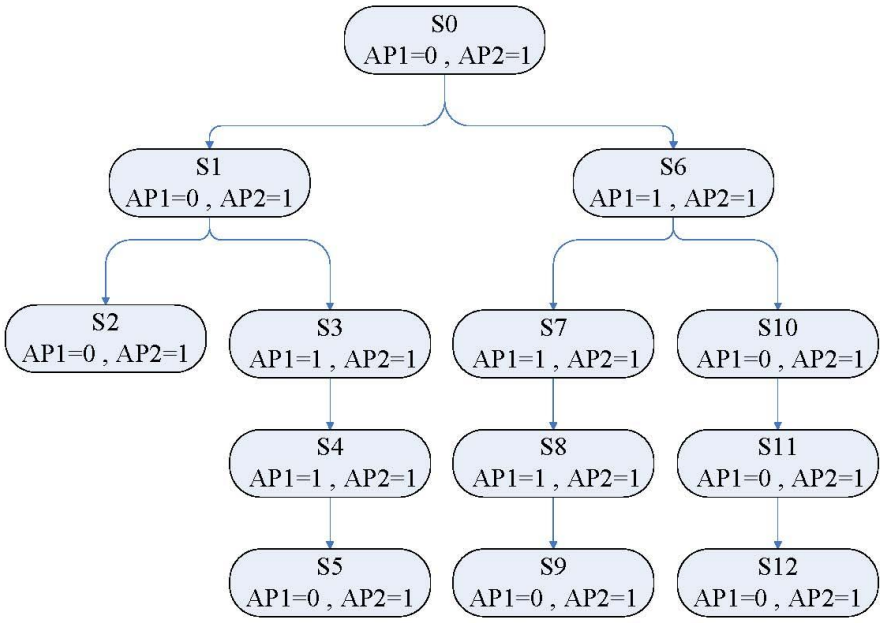
- Ambient logic formulas are decomposed into a CTL formula and a set of spatial formulas by formula reduction.
- Matching of an ambient topology and a spatial formula is a recursive procedure in which ambient topology nodes are assigned to formula nodes.
- Match process is successful when all nodes at ambient topology is matched to a spatial formula node.

Spatial Tree Matching



- Heuristic Functions. Heuristic functions are used at matching the *Parallel composition and Somewhere* connectives.
- The number of these trials are reduced by the help of auxiliary heuristic functions.

Generation of Kripke Structure



- Let AP be a non-empty set of atomic propositions. A Kripke structure is a four-tuple; M = (S, S0, R, L) where
- S is a finite set of states.
- $S0 \subseteq S$ is the set of initial states.
- $R \subseteq S \times S$ is a transition relation
- L: $S \rightarrow 2(AP)$ is a function that labels each state with the set of atomic propositions true in this state.

Complexity and Performance Analysis

- Time Complexity
- Space Complexity
- Performance Results

Time Complexity

- Where n is the number of capabilities in the ambient calculus specification, the time complexity of generating state transition system in worst case is $\sum_{k=0}^{n} \frac{n!}{k!}$
- The time complexity of checking spatial modalities are dependent to the type and number of the connectives of the spatial formulas. $O(a_{ne}^{(sw_w+not+d_w)})$
- The space complexity of the space generation is where *n* is the number of capabilities.

Performance Results

- As an example to performance results, a specification with 16 ambients and 37 capabilities generates nearly 630,000 states with memory consumption under 8 MB and a time of under 300 seconds.

Conclusions and Future Work

- Case studies and complexity analysis show that size of the state transition system is the most significant element at time and spatial cost of model checking.
- Number of states grows exponentially as capability numbers increase linearly.
- A partial order reduction might decrease the number of the states.
- We are currently develop a partial order reduction technique for ambient calculus.

Thank you for listening...

Questions?