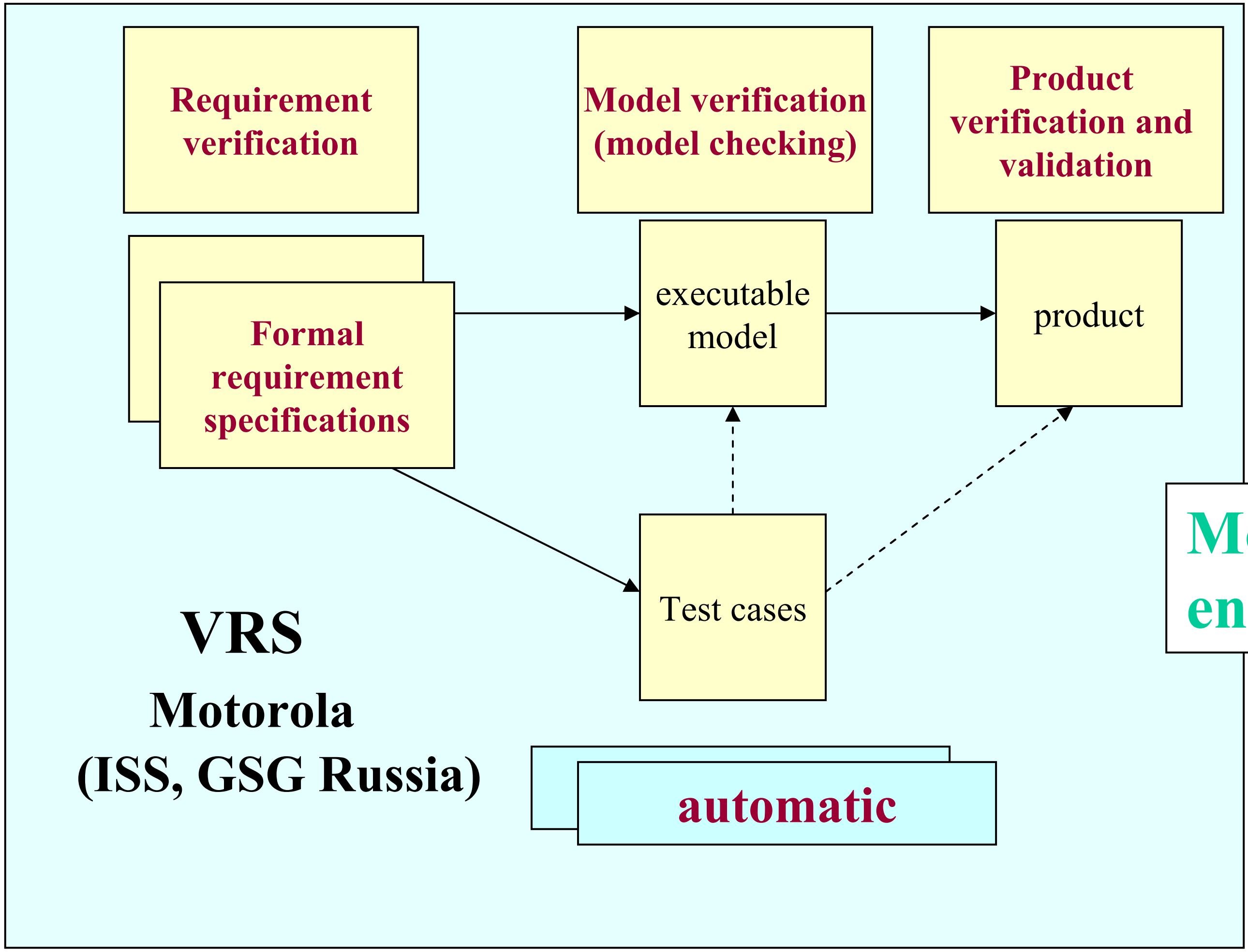
INTAS Timisoara
10-11 December 2006

Insertion modeling and requirement specifications for distributed concurrent systems

A.Letichevsky

Requirement specifications in design process



Model driven engineering

Insertion Modeling:

Developing and investigating of the models of distributed concurrent systems by means of representing them as a composition of interacting agents and environments

(A.Letichevsky, D.Gilbert, 1996)

Agents: **attributed transition systems**
Environments: **attributed transition systems with insertion function**
Composition: **continuous insertion function, characterizing the behavior of environment with inserted agents**

Insertion function

Agents and environments are considered up to bisimilarity and can be identified with their behaviors. *F*(*X*) is a complete behavior algebra over action algebra *X* (a kind of process algebra).

$$\mathrm{Ins}: E \times F(A) \to E,$$

$$E \subseteq F(C), A \subseteq C$$

$$\mathrm{Ins}(e,u) = e[u]$$

Multilevel insertion:
structure
mobility

$$e[u_1, u_2, \ldots, u_n] = (\ldots((e[u_1])u_2)\ldots)[u_n]$$

$$e[e_1[u_1], e_2[u_2], \ldots, e_n[u_n]]$$

$$(e[e_1[u_1], e_2[u_2], \ldots, e_n[u_n]])[v_1, v_2, \ldots, v_n]$$

Insertion equivalence

$$u \sim_E u' \Leftrightarrow \forall(e \in E)(e[u] = e[u'])$$

$$[u] : E \to E$$

$$u \sim_E u' \Leftrightarrow [u] = [u']$$

$$e([u] * [v]) = e[u, v]$$

Agents transform the behaviors of environment

Abstraction levels for insertion models

Abstract models

The states of agents and environments identified with their behaviors

Insertion functions – recursive definitions in behavior algebra, rewriting logic

Can be used for encoding CCS, CSP, ACP, π-calculus, mobile ambients etc.

Symbolic models

The states of environment with inserted agents are labeled by logic formulas over attributes of agents and environments or identified with such formulas

Concrete models

The states of agents identified with valuations or (partial) mappings from attributes (or attribute expressions) to their values (SDL,UML,…)

Abstract models

$$\frac{e \xrightarrow{a} e', u \xrightarrow{b} u', h(a,b,c)}{e[u] \xrightarrow{c} e'[u']}$$

$$(e+e')[u] = e[u] + e'[u], e[u+u'] = e[u] + e[u']$$

One step insertion rules

$$(a.e' + e'')[b.u' + u''] = c.e'[u'] + f, h(a,b,c)$$

Basic Protocols Specification Language

(Symbolic models)

BP specification:

Environment description (structural requirements)

Defines the signature and axioms of Basic Language
(first order logic language used for the labeling of environment states possibly with some temporal modalities for the past)

The set of Basic Protocols (local requirements)

Define the transitions of environment with inserted agents

Global requirements

Define the properties of a system in terms of temporal logic

Basic protocols

Combination of Hoare triples and insertion modeling

First order quantifiers over typed variables

$$\forall x(\alpha(x) \rightarrow < P(x) > \beta(x))$$

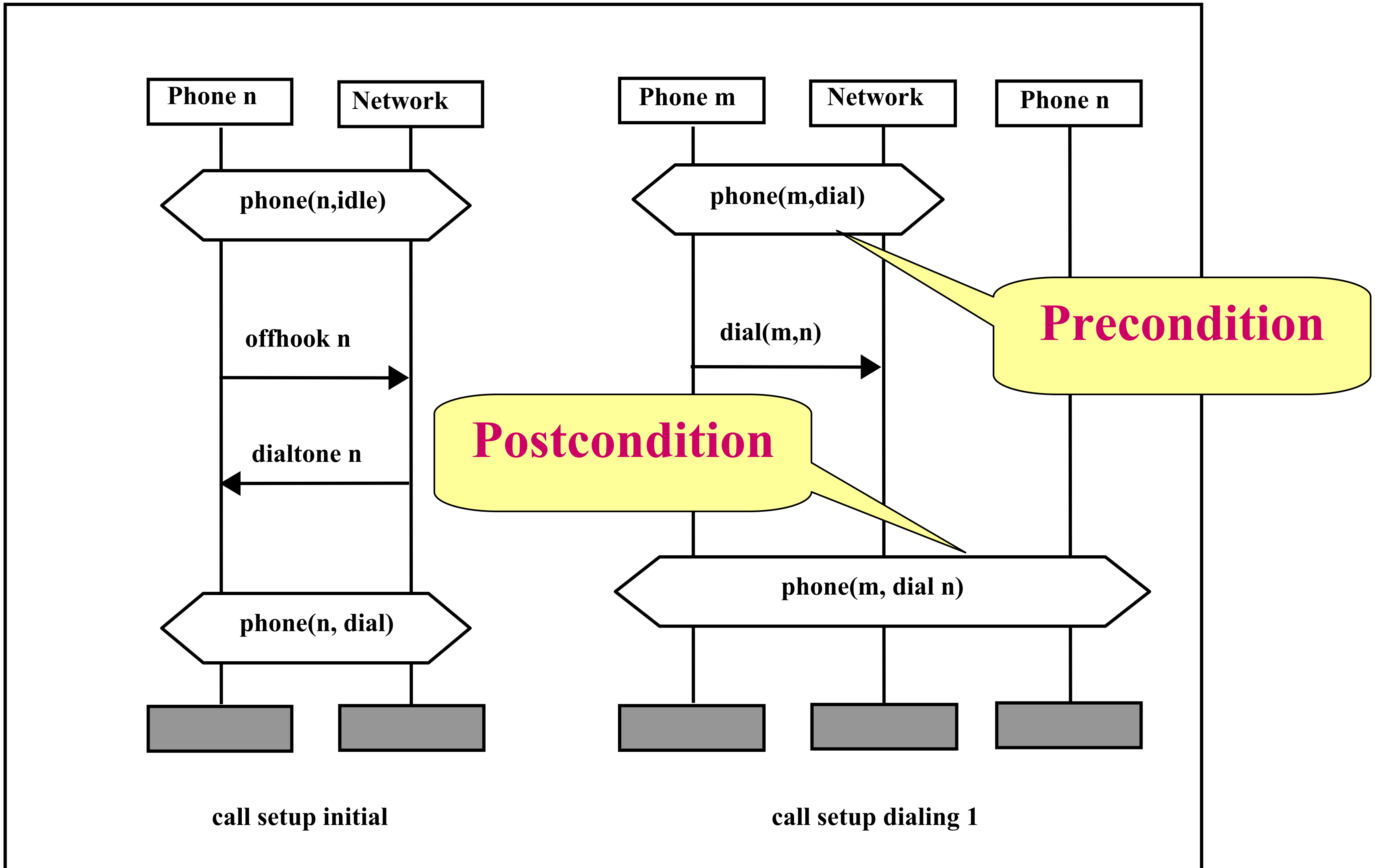
Precondition

Postcondition

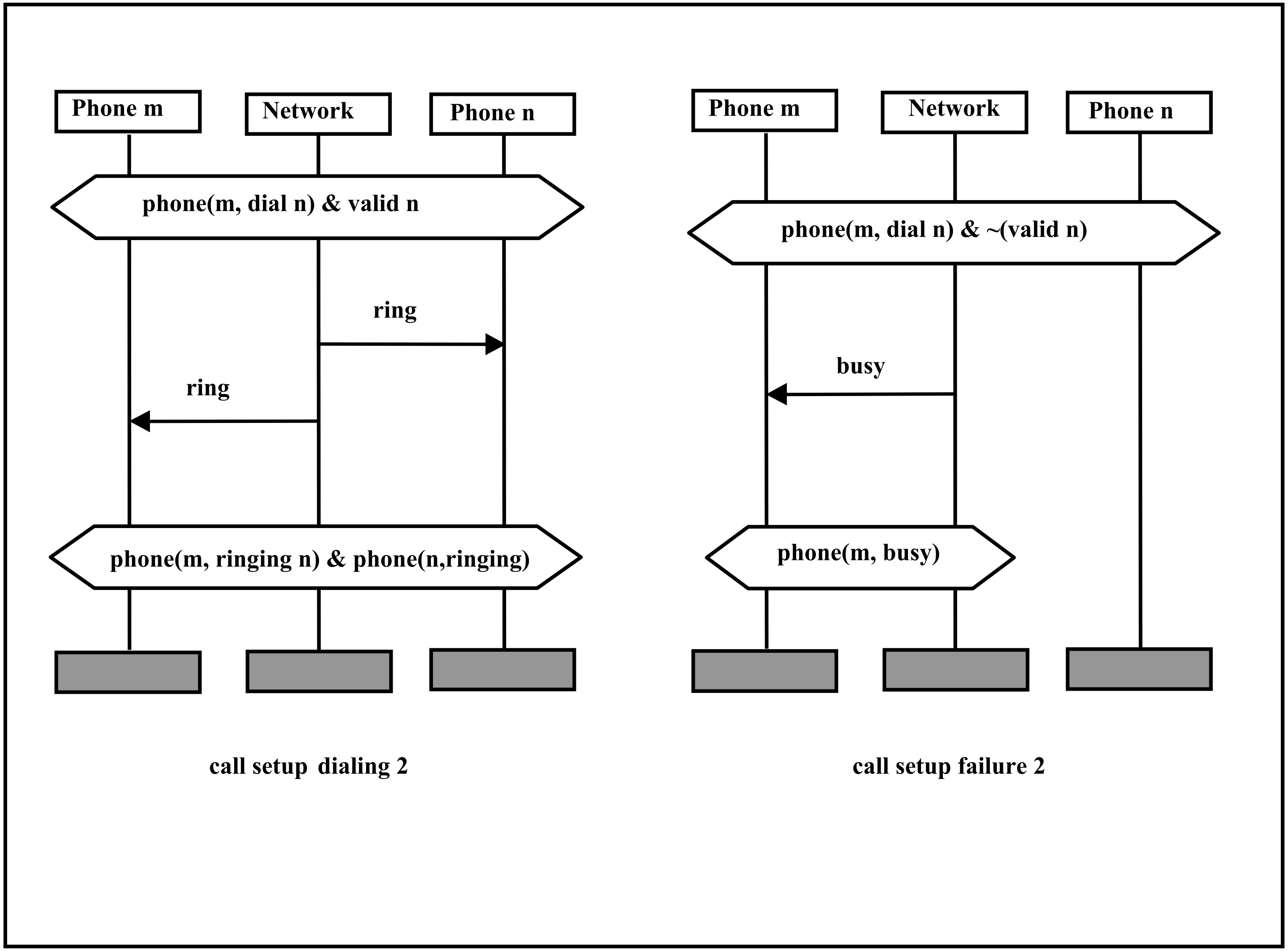
Finite process (behavior) of attributed environment with inserted agents

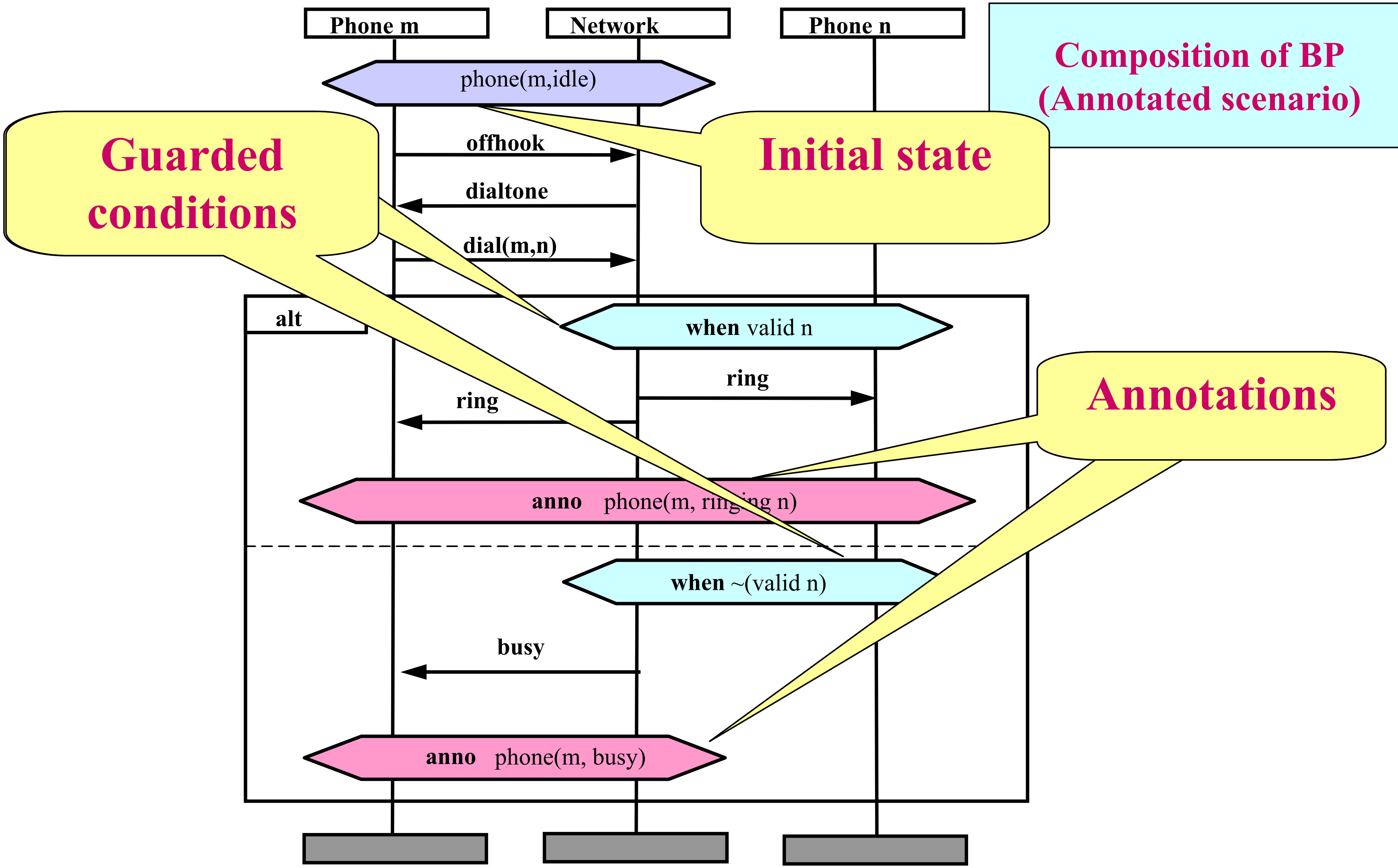
Properties of environment

Two basic protocols for telecommunication example



Two more protocols





Composition of BP
(Annotated scenario)
Phone m
Network
Phone n
phone(m,idle)
Initial state
Guarded conditions
offhook
dialtone
dial(m,n)
alt
when valid n
ring
ring
Annotations
anno phone(m, ringing n)
when ~(valid n)
busy
anno phone(m, busy)

The use of basic protocols

Formalizing requirements

Experience in Telecommunications,
Telematics and other application domains

VRS

Verification of Requirement Specifications
a tool developed by ISS for Motorola

Static requirements checking

Dynamic requirements checking

(projects for Motorola)

Proving correctness of parallel programs based on MPI and OpenMP

(new projects for Intel)

Generating tests from requirement specifications

Static requirements checking

Disjunction of preconditions is valid

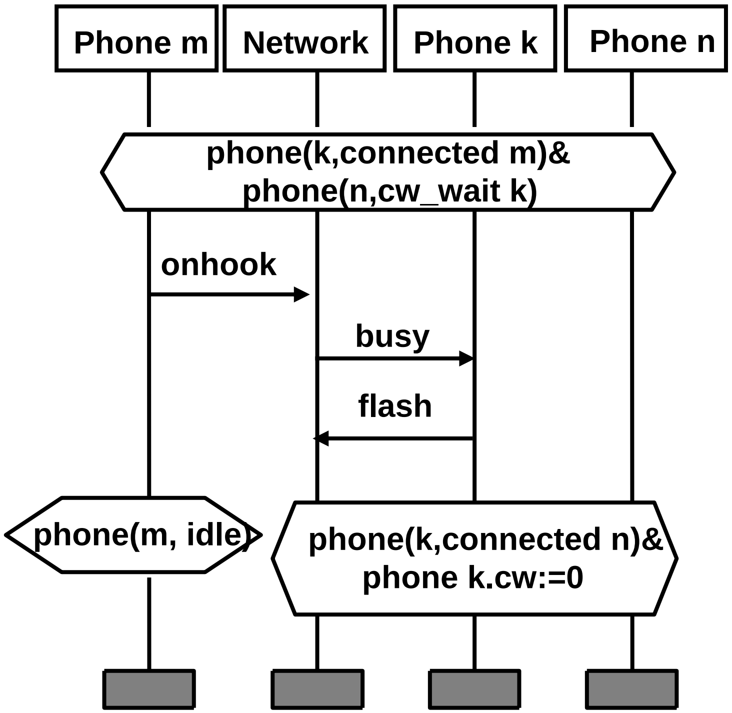
- **Proving consistency and completeness**
- **Proving safety**
- **Computing invariants**

Preconditions for BPs (with the same external actions) must not intersect

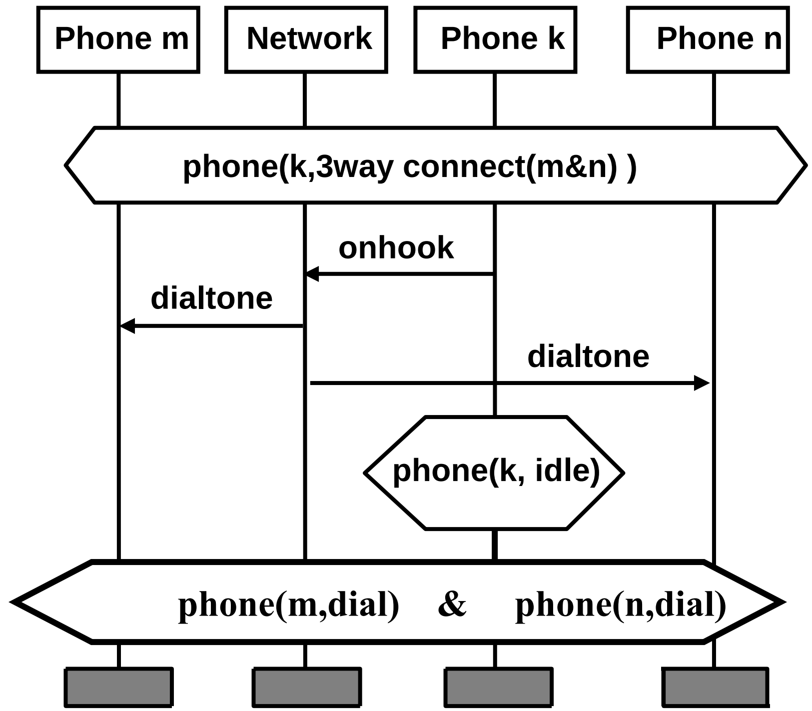
Dynamic requirements checking

- **Symbolic model checking with deduction for abstract models**
- **Checking safety and reachability**
- **Generating traces and checking properties for concrete models**

Inconsistent protocols
(inconsistency of features 3way Calling and Call Waiting)

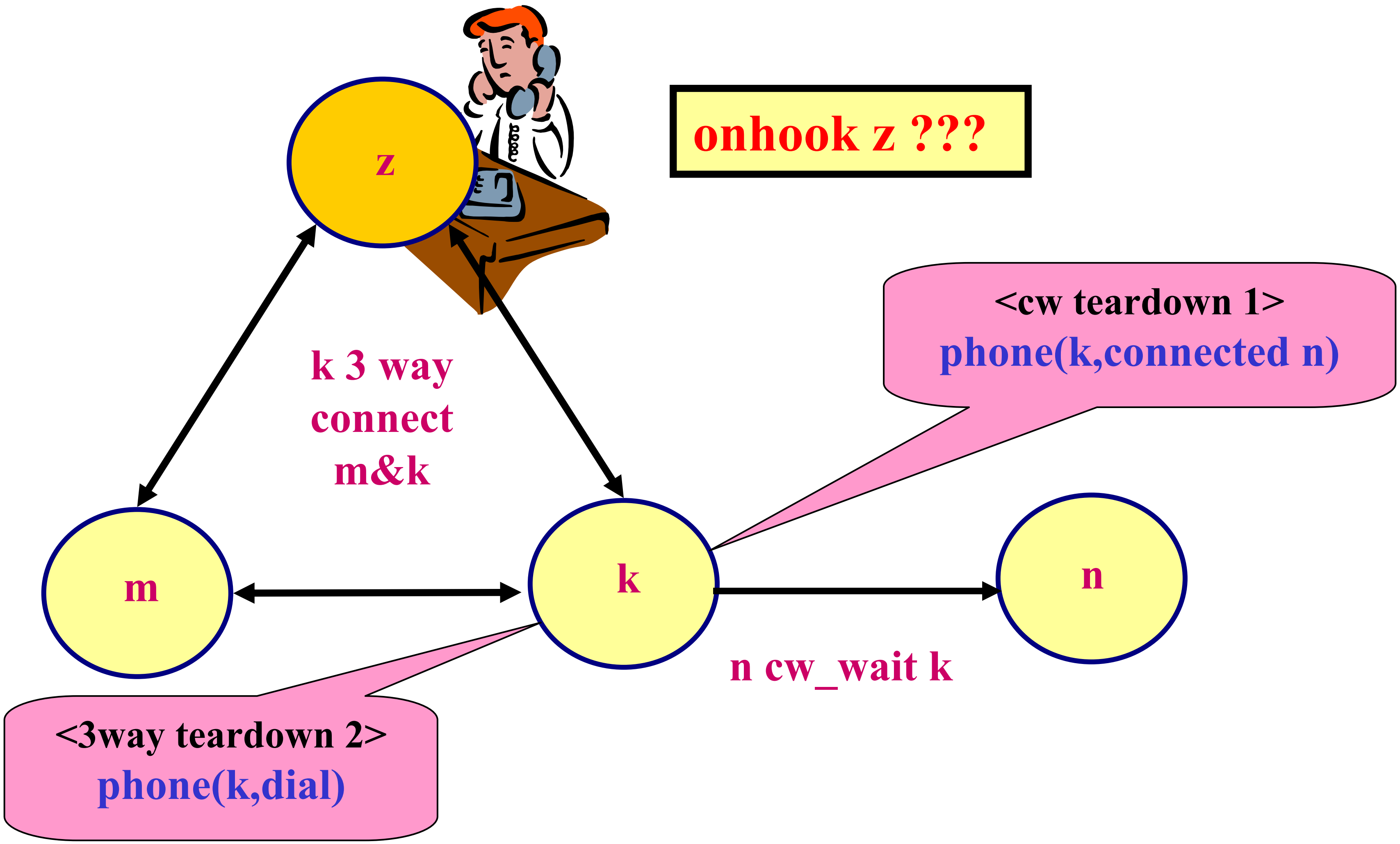


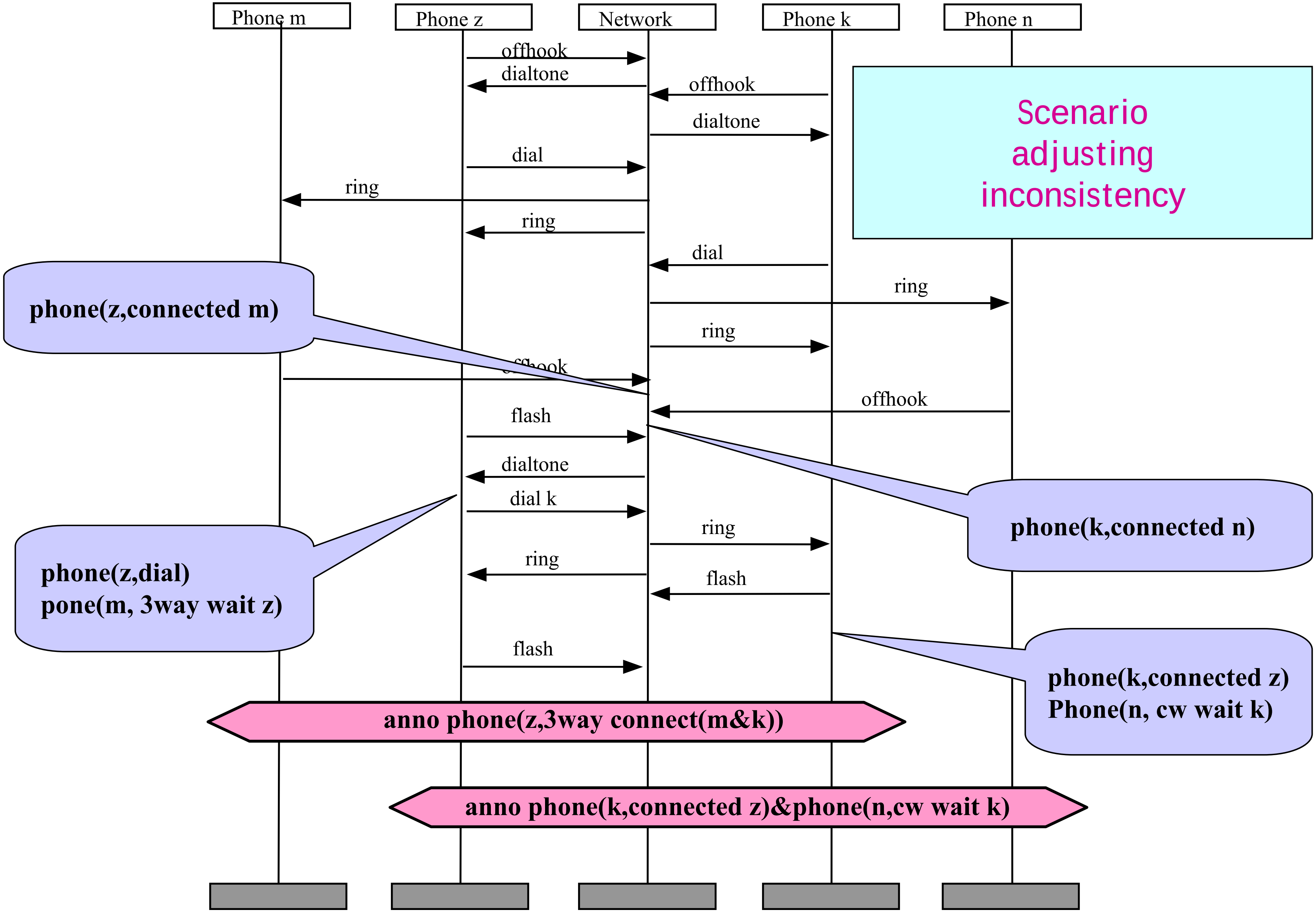
Protocol cw teardown 1



Protocol 3way teardown 2

Inconsistent state




Phone m
Phone z
Network
Phone k
Phone n
Scenario adjusting inconsistency
offhook
dialtone
offhook
dialtone
dial
ring
ring
dial
ring
ring
phone(z,connected m)
offhook
offhook
flash
dialtone
dial k
ring
ring
flash
flash
phone(k,connected n)
phone(z,dial)
pone(m, 3way wait z)
phone(k,connected z)
Phone(n, cw wait k)
anno phone(z,3way connect(m&k))
anno phone(k,connected z)&phone(n,cw wait k)

Basic Language

Signature

Data structures: types, functions.

Attributes: distinguished functional symbols (simple and parameterized attributes)

Agent attributes: $m.g(x,y,\ldots)$

Predicates: interpreted (for example, numeric) and noninterpreted

Special types:

agent types, agent names (ids), age … real arithmetic), enumerated, …

(state assertions like **state (Phone** *n*) = idle)

More details and concrete syntax depend on subject domain

Axioms and algorithms for validity of formulae (calculus).

The language of preconditions: first order formulae of BL.

The language of postconditions: the same as preconditions + assignments and other imperative expressions).

$$(x{:=}y) \sim (x'{=}y)$$

Validity relation

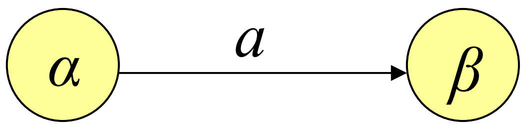
$$s \models \alpha, \alpha \in \mathbf{BL}$$

For states labeled by formulas

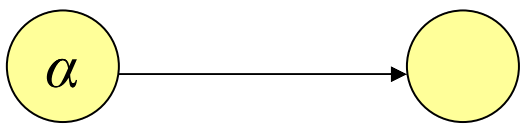
$$s \models \alpha \Leftrightarrow (s = (\gamma : t)) \wedge (\gamma \models \alpha)$$

Process language

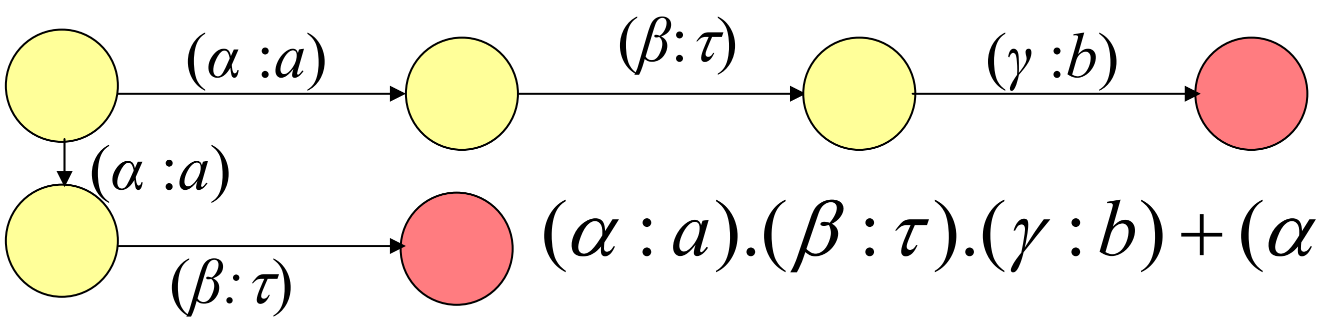
User languages: MSC, annotated MSC, SDL, UML, …
Semantics: attributed transition systems and their behaviors



Behaviors of attributed transition systems are **attributed behaviors**



$$\alpha : (a.(\beta : \gamma : b) + a.(\beta : \Delta))$$



$$(\alpha : a).(\beta : \tau).(\gamma : b) + (\alpha : a).(\beta : \tau)$$

Concrete implementation of systems of BPs

Concrete attributed transition system S implements the set of BPs if for each BP

$$\forall x(\alpha(x) \to < P(x) > \beta(x))$$

$$s \models \alpha(x) \to \mathbf{beh}(s) = (P(x);(\gamma : \Delta)) * Q + R, \gamma \models \beta(x)$$

* is a partially sequential composition to be defined later
Q and R are also to be defined

Questions

BPS define abstractions for their concrete implementations.
Studying of BPS we study also their concrete implementations

- **What is abstraction?**
- **What is abstract implementation?**
- **What is concrete implementation?**
- **What are the relations between abstract and concrete implementations?**

Main result

Systems S_P and S^P are attributed systems with states labeled by the statements of BL.

They define semantics of BPS

K(P) is a class of concrete implementations of P.

Theorem

System S_P(S^P) is a direct (inverse) abstraction of any concrete implementation of a system P of basic protocols from the class K(P)

Abstraction relation on states

The same attribute labeling and validity

$\mathbf{Abs} \subseteq S \times S'$

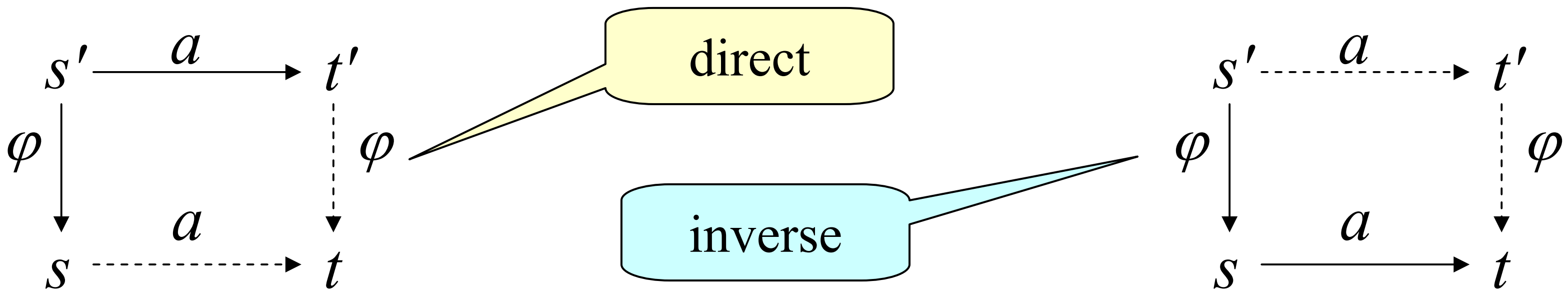
$(s, s') \in \mathbf{Abs} \Leftrightarrow \forall(\alpha \in \mathbf{BL})((s \models \alpha) \Rightarrow (s' \models \alpha))$

more abstract: $s \lhd s'$

Abstraction relation on systems

$$S \lhd S' : \ \exists \varphi \subseteq \mathbf{Abs}^{-1} \qquad \lhd_{dir}, \lhd_{inv}$$

Preserve initial states



$$\forall (s \in S, s' \in S')((s', s) \in \varphi \wedge s' \xrightarrow{a} t' \Rightarrow \exists (t \in S)(s \xrightarrow{a} t \wedge (t', t) \in \varphi))$$

$$\forall (s \in S, s' \in S')((s', s) \in \varphi \wedge s' \rightarrow t' \Rightarrow \exists (t \in S)(s \rightarrow t \wedge (t', t) \in \varphi))$$

$$\forall (s \in S, s' \in S')(s \lhd s' \wedge s \xrightarrow{a} t \Rightarrow \exists (t' \in S)(s' \xrightarrow{a} t' \wedge t \lhd t'))$$

$$\forall (s \in S, s' \in S')(s \lhd s' \wedge s \rightarrow t \Rightarrow \exists (t' \in S)(s' \rightarrow t' \wedge t \lhd t'))$$

Direct and inverse abstractions

If some property is reachable in a system then it is reachable in *a direct abstraction* of the system.
Therefore: use direct abstraction for *verification* (safety condition for example)

If some property is reachable in *an inverse abstraction of a system* then it is reachable in the system itself.
Therefore: use inverse abstraction for *test generation* (reachability of error condition)

Abstract implementations of systems of basic protocols

- **Basic protocols: attributed systems labeled by pre- and postconditions;**
- **States identified with their state labels (formulas);**
- **Predicate transformer defines transitions;**
- **Partially sequential composition of behaviors defined by**
  - **Permutability relations on the set of actions.**
- **Direct and inverse implementations of BPs by systems S_P and S^P.**

Predicate transformers

$$\mathbf{pt}(\gamma, \beta) = \gamma', \gamma' \rightarrow \beta$$

Instanciated BP

$$\alpha \rightarrow < P > \beta$$

$$\gamma \rightarrow \alpha, \gamma \rightarrow < P > \gamma', \gamma' = ?, \gamma' \rightarrow \beta,$$

$$\gamma' = \mathbf{pt}(\gamma, \beta)$$

Monotonicity:

$$(\gamma \rightarrow \gamma') \rightarrow (\mathbf{pt}(\gamma, \beta) \rightarrow \mathbf{pt}(\gamma', \beta))$$

To compute **pt**(γ, β) = γ'

1. Reduce A to minisphere form and then to dnf

$$\gamma = \gamma_1 \vee \gamma_2 \vee \dots$$

$$\gamma_i = \gamma_{i1} \wedge \gamma_{i2} \wedge \dots$$

2. Delete all γ_{ij} such that

$$Attr(\gamma_{ij}) \cap Attr(\beta) \neq \emptyset$$

$$\gamma' = \gamma'' \wedge \beta$$

Example of predicate transformer

Permutability relation

Defined on the set of labeled actions
Transferred to pairs behavior-action

$$\neg((\alpha:\perp)\leftrightarrow b),\ \neg((\alpha:0)\leftrightarrow b)$$

$$\neg(u\leftrightarrow(\alpha:\tau))$$

$$(\alpha:\Delta)\leftrightarrow b\Leftrightarrow(\alpha:\tau)\leftrightarrow b$$

$$u+v\leftrightarrow b\Leftrightarrow u\leftrightarrow b\wedge v\leftrightarrow b$$

$$a.u\leftrightarrow b\Leftrightarrow a\leftrightarrow b\wedge u\leftrightarrow b$$

Monotonicity:

$$(\gamma\rightarrow\gamma')\wedge(\gamma:u\leftrightarrow a)\rightarrow(\gamma':u\leftrightarrow a)$$

Partially sequential composition

$$u = \sum_{i \in I} a_i . u_i + \varepsilon_u, \; v = \sum_{j \in J} b_j . v_j + \varepsilon_v$$ Canonical form of behaviors

$$u * v = \sum_{i \in I} a_i . (u_i * v) + \sum_{u \leftrightarrow b_j, j \in J} b_j . (u * v_j) + (\varepsilon_u ; \varepsilon_v)$$

$$(\Delta ; \varepsilon) = \varepsilon, \; (\perp ; \varepsilon) = \perp, \; (0 ; \varepsilon) = 0$$

$$((\alpha : \varepsilon) ; \varepsilon') = \alpha : (\varepsilon ; \varepsilon')$$

Abstract implementation

$$P(\alpha) = \{p \in P_{inst} \mid \alpha \to \mathbf{pre}(p)\},$$

$$P(\alpha) = \{p \in P_{inst} \mid \neg \models \neg(\alpha \wedge \mathbf{pre}(p))\}$$

$$\mathbf{T}(\alpha, p) = \mathbf{pt}(\alpha, \mathbf{post}(p))$$

Instantiated BPs

Terminal protocols

$$\mathbf{S}_\alpha^\infty = \sum_{p \in P(\alpha)} \mathbf{proc}(p) * (\mathbf{T}(\alpha, p) : \Delta) * \mathbf{S}_{\mathbf{T}(\alpha,p)}^\infty)$$

$$P_\alpha = P_\alpha^0 \cup P_\alpha^1$$

$$\mathbf{S}_\alpha = \sum_{p \in P^1(\alpha)} \mathbf{proc}(p) * (\mathbf{T}(\alpha, p) : \Delta) * \mathbf{S}_{\mathbf{T}(\alpha,p)} + \sum_{p \in P^0(\alpha)} \mathbf{proc}(p) * (\mathbf{T}(\alpha, p) : \Delta) * (\mathbf{S}_{\mathbf{T}(\alpha,p)} + \Delta)$$

Composition of two BPs (simple scenario)

$$\mathbf{proc}(p) * (\mathbf{T}(\alpha, p) : \Delta) * \mathbf{proc}(q) * (\mathbf{T}(\mathbf{T}(\alpha, p), q) : \Delta)$$

Concrete implementations

- **Environment state**
- **Insertion function (transitions)**

The structure of a concrete implementation K

BL is interpreted on a concrete multisorted algebraic system.
The signature of K is extended by hidden attributes and symbols.
The states of environment:

$$s[q_1 * \ldots * q_m][u_1,\ldots,u_n]$$

s is the mapping from attribute expressions to their values.
$\boldsymbol{q_1,\ldots,q_m}$ partially sequential composition of BPs.
$\boldsymbol{u_1,\ldots,u_n}$ the states of named passive agents (they do not participate in protocols).
Special attributes: ActiveBP (the list of active protocols**)**, *b*.**active** (the list of active agents**).**
Environment actions: start, start *b*, **terminate** *b*

$$(\gamma : \mathbf{start}) \leftrightarrow a \Leftrightarrow (\gamma : \mathbf{terminate}\ b) \leftrightarrow a \Leftrightarrow (\gamma : \Delta) \leftrightarrow a$$
$$\neg(u \leftrightarrow (\gamma : \mathbf{start})), \neg(u \leftrightarrow (\gamma : \mathbf{terminate}\ \ b))$$

Initial states of environment:

$$s[\gamma : \mathbf{start}][m_1 : u_1, \ldots, m_k : u_k]$$

γ **is the conjunction of equalities for *s*.**

The state of successful termination:

$$s[\gamma : \mathbf{start}][\Delta]$$

$$s[q_1 * \ldots * q_m][u_1, \ldots, u_n]$$

Transitions:

- **The change of a state of a protocol;**
- **The termination of a protocol;**
- **The launching of a new protocol;**
- **The termination of a system.**

Insertion function

Transition of BP

$$\frac{s \xrightarrow{(\beta:a)} s', q \xrightarrow{(\beta:a)} q', s \models \beta}{s[q] \xrightarrow{(\beta:a)} s'[q'][m_1 : u_1, \ldots, m_k : u_k]}$$

Participate in q, but not in q'

Termination of BP

$$\frac{s \models \gamma, s \xrightarrow{\textbf{terminate } b} s'}{s[(\gamma : \textbf{terminate } b) * q] \xrightarrow{(\gamma:\tau)} s'[q][m_1 : u_1, \ldots, m_k : u_k]}$$

$$\frac{s \models \gamma, s \xrightarrow{\textbf{terminate } b} s'}{s[\gamma : \textbf{terminate } b] \xrightarrow{(\gamma:\tau)} s'[\gamma : \textbf{start}][m_1 : u_1, \ldots, m_k : u_k]}$$

Participated in b

Launching BP

$$\frac{\gamma \models \mathbf{pre}(b') \wedge \beta, \mathbf{proc}(b') = (\beta : a).p + p', q \leftrightarrow (\beta : a), s \xrightarrow{\mathbf{start}\, b'} s' \xrightarrow{(\beta:a)} s''}{s[q * t][m_1 : u_1, \ldots, m_k : u_k] \xrightarrow{(\beta:a)} s''[q * t * p * t']}$$

$$\frac{\gamma \models \mathbf{pre}(b') \wedge \delta, \mathbf{proc}(b') = p + (\delta : \Delta), s \xrightarrow{\mathbf{start}\, b'} s'}{s[q * t] \xrightarrow{(\delta:\tau)} s''[q * t * t']}$$

$$t = (\gamma : \mathbf{start}), (\gamma : \mathbf{terminate}\, b),\ \ t' = (\mathbf{pt}(\gamma, \mathbf{post}(b')) : \mathbf{terminate}\, b')$$

Termination of a system

$$\frac{s \models \gamma, s \xrightarrow{\mathbf{terminate}\ b} s'}{s[(\gamma : \mathbf{terminate}\, b)] \xrightarrow{(\gamma:\tau)} s'[\gamma : \mathbf{start}]}$$