Overview of Name Model Lectures

1. Roles of Name Models
    - Definitions
    - Model of Communication
    - Models of Computation
2. A Taxonomy of Name Models
    - Lattice Axes
    - An Instance of the Taxonomy
3. Associative Broadcast Model of Parallel Programming

Naming Systems

1. Objects - Entities which have a name(s) and a state

2. Name - A symbol or an expression which evaluates to an object.

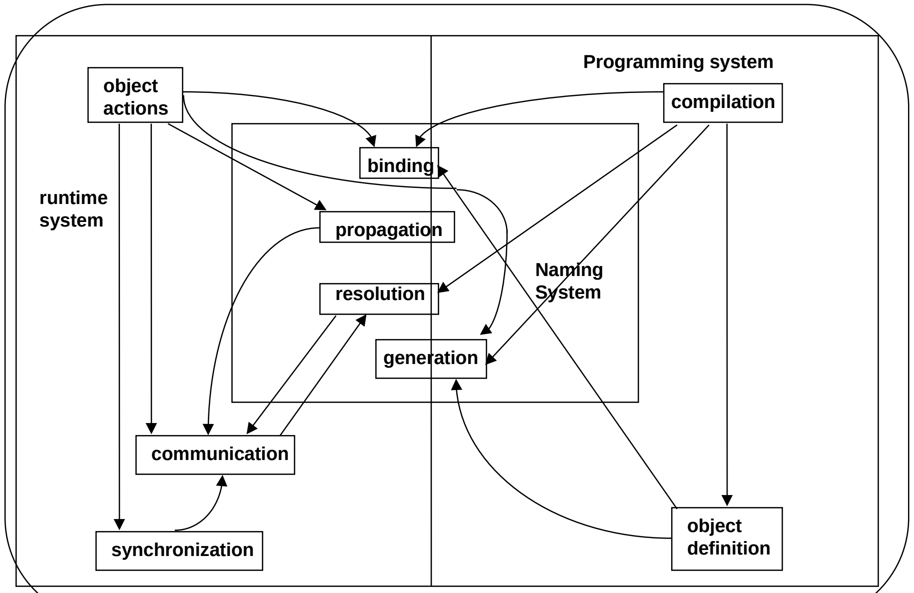
3. Binding - An algorithm which assign names to objects.

4. Resolution - An algorithm which resolves names to objects

5. Generation - An algorithm for creating names to be assigned to objects.

7. Context - The set of names known to an object at a given time.

8. Propagation - Algorithm for moving names among objects.



Roles of a Naming System

Naming Systems - Resolution versus Binding

binding - Assignment of a name to an object or set of objects.

binding => Defines semantics of naming systems

analogy => Language definition which assigns operators to syntactic constructs

resolution - Takes a syntactic construct (a name) and determines the the set of objects to which the name is bound.

resolution - Defines the syntax of a naming model

analogy => compilation of a language which determines which operations have been associated with a syntactic construct.

Distributed Dynamic Resources and Objects have rekindled interest in naming models and systems

dynamic sets of network nodes
peer to peer network architectures
dynamic video conferencing
fault-tolerance and reconfiguration
wireless communication
mobile computing

These systems require interaction among dynamic sets of entities and continuous availability. The naming models of a programming system determine the communication properties of the systems which are implemented.

Why study naming systems?
and
Why study binding in particular?

1. to be able to design naming systems with desired properties.
   example - be able to formulate a resource management system without polling (scalability, availability)

2. to be able to compare and evaluate naming systems for their properties and capabilities
   a) What class of algorithms can be directly implemented in a model of communication using a given name model?

   b) Complexity of expression of given algorithm within a given name model.
      (consider the readers and writers problem with priority for writers and strong fairness in Ada.)

Binding and Communication

Binding defines the allowed set of direct communications among objects

1. A message is sent to a name

2. An object can therefore only communication with the other objects for which it has names.

Types of Names

1. A Direct name is bound to set of terminal objects for a communication

2. Indirect name - bound to an object which purpose is to support procedural implementations of communications - channel, buffer, etc.)

Procedural Versus Functional or Implementation of Communication Operations

Functional Specification - the communication is invoked by an operation which is atomic in the specification language.

Sending to a set of recipients by an atomic broadcast operation

Procedural Specification - the communication operation is realized by a program in a language at lower level of abstraction.

Sending to a list of recipients by looping over a send() operation.

Procedural Versus Functional or Direct Implementation of Communication Operations

Why should anyone care?

1. Compactness of algorithms

2. Properties of Solutions- fully distributed, fully symmetric,
minimal algorithms
(FDFSMC)
\- availability

3. Efficiency of Execution - the higher the level of abstraction of an IMPLEMENTATION the lower the efficiency

Example: Polling solutions versus direct solutions

Polling Solutions for Communication Operations

Push Polling - A sender sends repeatedly until a response is received.

Eg - A client request to a server is repeated until the request is granted.

Pull Polling - A receiver cycles across a set of “ports” until it receives a desired message.

Eg. - A server examines its message queue periodically to look for different classes of messages.

Models of Parallel Computation

A parallel computation is a partial order over events.

event - send/receive message
- beginning/end of a computation
- binding/unbinding of name to object

causalty among events

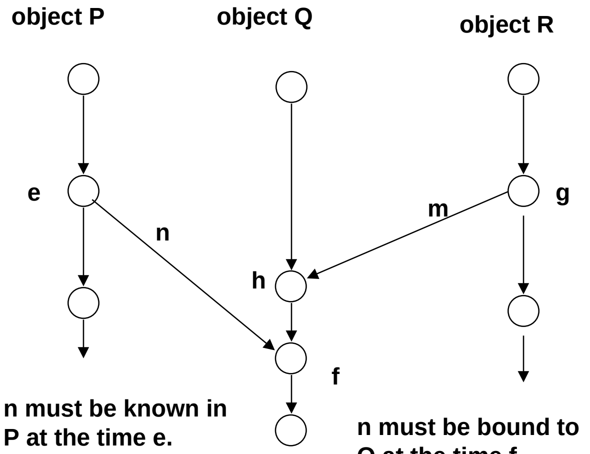
e1 |-> e2 : e1 must precede e2

e1 |-> e2 ; e2 |-> e3 : then e1 |->e3

e1 |-> e2 ; e3 |-> e4 ; then e1 and e2 are not ordered

Model of choice for reasoning about communication systems

An example of communication in a computation



(e,n,f) and (g,m,h) communications.

e,f,g and h are events

e|-> f , g|->h.

n must be known in P at the time e.

n must be bound to Q at the time f.

Attributes of Name Models - Axes of a Taxonomy

1. Dynamic/Static Binding of Names to Objects
2. Dynamic/Static Object Domains
3. Multiplicity of Names for Given Object within a Single Domain
4. Aliasing of Names
5. Sharing of Names Across Domains
6. Descriptive Names

A descriptive name consists of a selection of attribute name/value pairs.
The allowed set of names defines a communication domain.
The range of name/attribute value pairs defines set of names admissible in the communication domain.

Lattice Taxonomy of Naming Models

1. Define a lattice where each property defines a coordinate of unit length with cap letter property at one end of axis and lowercase letter property at other end of axis.

2. Every naming model must fall at a vertex of the lattice.

3. Every cap letter property enables more direct (non-procedural) communication operations than the corresponding lower case property.
(This is not an includes and enlarges relationship but only an enlarges relationship because of interactions between name models and other representation features.)

4. Therefore the lattice has a natural increasing ordering from lowercase to cap directions

---

Properties of Name Binding Systems

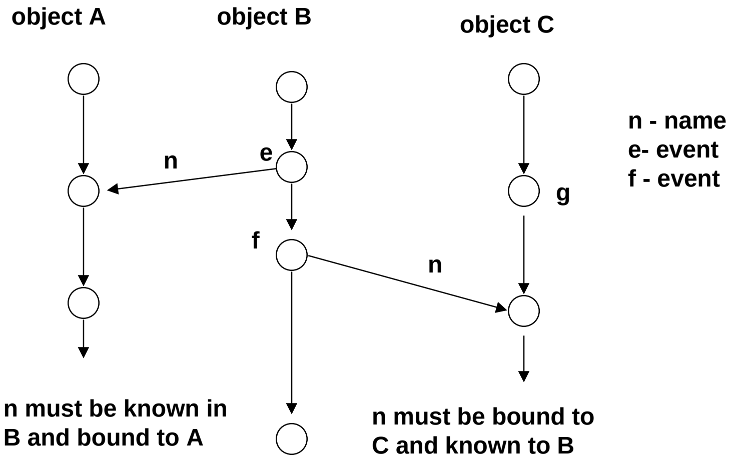
1. Dynamic/Static binding

- dynamic implies that a name may be bound to multiple objects over the lifetime of a single computation. Indirection is direct! - B

- static implies that procedural implementation of indirection is required. - b

2. Dynamic/static domains - dynamic implies that the set of names known to an object may vary over the lifetime of a computation. (D/d)

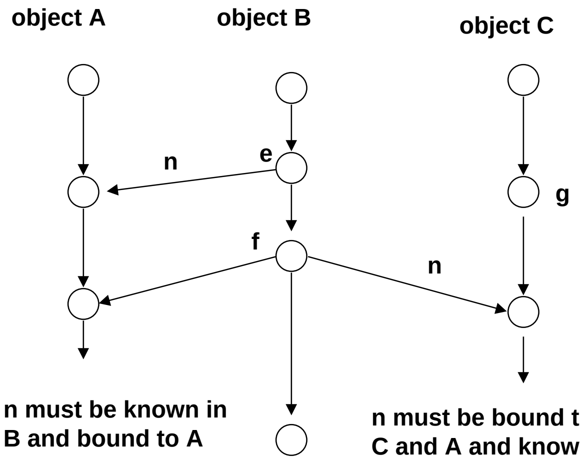
Examples of Dynamic Bindings



n must be known in B and bound to A at the time of e

n must be bound to C and known to B at the time of f

Examples of Dynamic Bindings



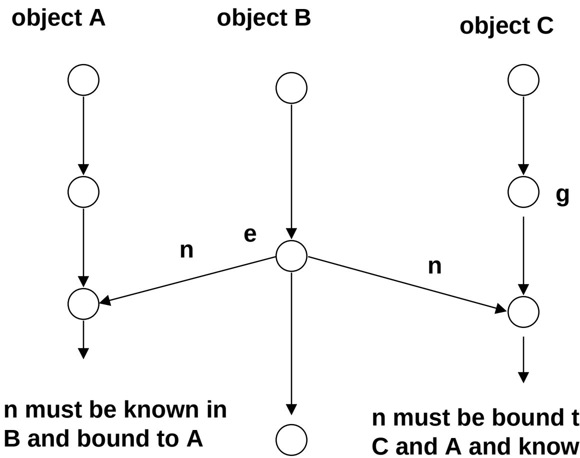
n - name
e- event
f - event

n must be known in
B and bound to A
at the time of e

n must be bound to
C and A and known to B
at the time of f

---

3. Multiplicity - A name (can,M) (cannot,m) be bound to multiple objects.-



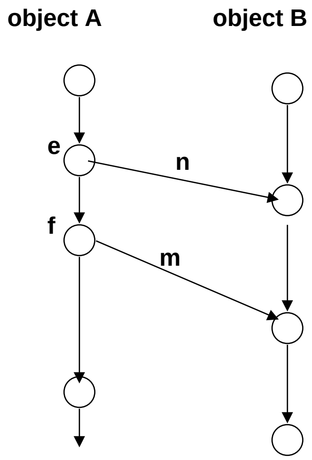
n - name
e- event
f - event

n must be known in
B and bound to A
at the time of e

n must be bound to
C and A and known to B
at the time of e

---

4. Aliased names - given object refers to another given object through multiple names - (A/a)



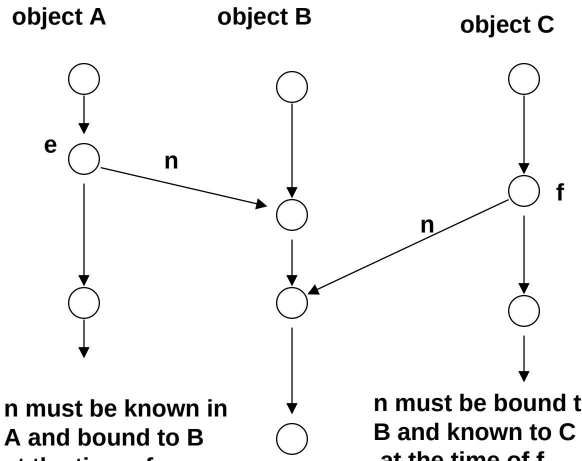
n - name
e- event
f - event

n must be bound to
B and A and known to B
at the time of e

B must be bound to
B and A and known to B
at the time of f

---

5. Shared Names - the name bound to a given object appears in multiple domains (S/s)



n - name
e- event
f - event

n must be known in
A and bound to B
at the time of e

n must be bound to
B and known to C
at the time of f

Properties of Name Binding Models

6. Descriptive Reference - state-based names: The names bound to an object directly reflect its internal state. (R/r)

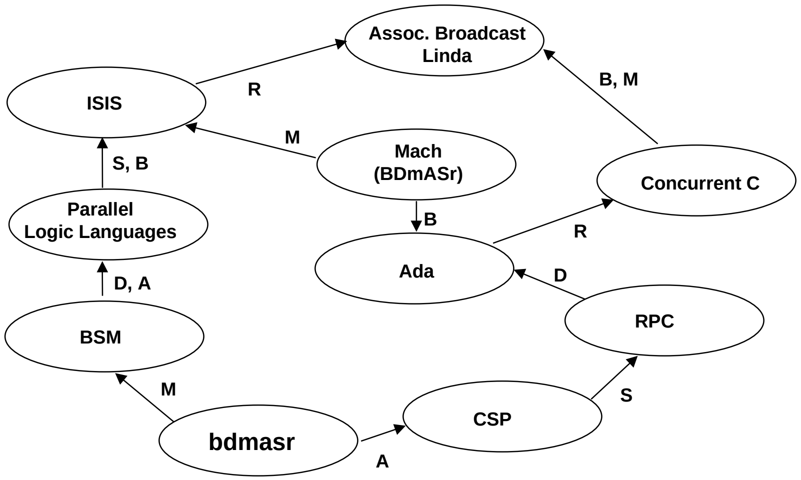
   a) Each identifier has a semantic interpretation
   (Eg. “This process is waiting for resource x.”)

   b) The name model must include provision for dynamic binding of names to objects to reflect externally interesting changes in object state.

Associative Broadcast Implementation of Descriptive Reference

1. Define a set of attributes characterizing a set of objects.
   attributes - enumerated types
2. Define for each object instance a “profile” selected from among those attributes which characterize its state.
   profile - set of name/value pairs from the attributes
3. Each message consists of a “selector” and data.
   selector - a conditional expression over the attributes.
4. All object instances are in a common broadcast domain.
5. Messages are received by those objects where the values in the profile causes the selector to evaluate to “true.”
6. Each object maintains its own profile.
7. Objects follow a task model of execution.

Naming Systems



Representations of the Taxonomy

Associative Interfaces

Associative Interfaces include:
- * An “accepts” interface which specifies the interactions in which a component model will accept
- * A “requests” interface which specifies the interactions a component model will initiate

An “interaction” includes:
- * Acceptance of a message
- * Execution of a unit of work
- * Propagation of the interaction

Associative Interfaces

“Accepts” Interface

a) profile - compositional element of interface; specifies what interactions will be accepted.

profiles - name/value pairs from a set of attributes that define and characterize the services offered by the component

b) transaction - defines the interaction, including the data or parameters associated with each interaction

c) protocol - sequences steps of an interaction.
Call-return, data flow, etc.

Associative Interfaces

“Requests” Interface

a) selector expressions - compositional element of interface; specifies what services are desired

selector - an expression in propositional logic over the attributes of the domain models which evaluates to true for certain values of the attributes

b) transaction - defines the interaction, including the data or parameters necessary for a given interaction

c) protocol - specifies sequence of interactions

Associative Interactions - Composition

Composition - Matching of “requests” to “accepts” or selectors to profiles

- A composition occurs (interaction occurs) if a selector evaluates to “true” for the attribute value pairs of a profile.

Each matching establishes a relation among component models, for both components, and defines the transaction that flows between the component models and the protocol to sequence the flow of transactions. (Transactions carry the data and parameters.)

Mechanisms for resolving the selectors with the current set of profiles posted by the components in the domain of the model

a) Compile Time Composition - selectors (which are names) are matched with profiles (which also are names and perhaps typed parameters) by a compiler

b) Link Time - selectors are names and profiles are names

c) Initialization Time - selectors are “templates” and profiles are name/value pairs of attributes but resolution is done once and the set of interactions is static across the execution

d) Run Time - selectors are “templates” and profiles are name/value pairs of attributes. Selectors and profiles are varied during an execution and resolution creates an interaction

Associative Interactions - Associative Broadcast Communication

0. Assume the communication medium is a timed reliable broadcast.

1. Each object maintains an accepts interface which defines the messages it wants to receive.

2. Each message consists of a selector expression and a message.

3. Messages are received by those objects whose profile causes the selector expression to evaluate to true.

4. Each object maintains its own profile.

5. Therefore communication is over dynamic sets.

6. Objects can enter and leave a communication group without executing a group membership protocol.

Associative Interfaces and Programming of Nomadic Applications

What is a "Nomadic Application?"

Distributed system where group memberships are dynamic. Systems enter and leave an interacting group.

Tasks reading and writing replicated data objects.

Group membership is automatically dynamic when interactions are through associative broadcast.

Formulating distributed applications in associative broadcast communications with a timed reliable broadcast medium leads to very simple algorithms and program structures for dynamic distributed systems.

Summary

Formulation of systems in terms of associative interactions among components:

1. Integrates compilation and communication
2. Integrates compile time and run time program composition.
3. Distributed systems formulated in associative interactions are naturally dynamically structured.

There are many possible applications of associative interactions.