Building Multifunctional Knowledge-Bases from Components

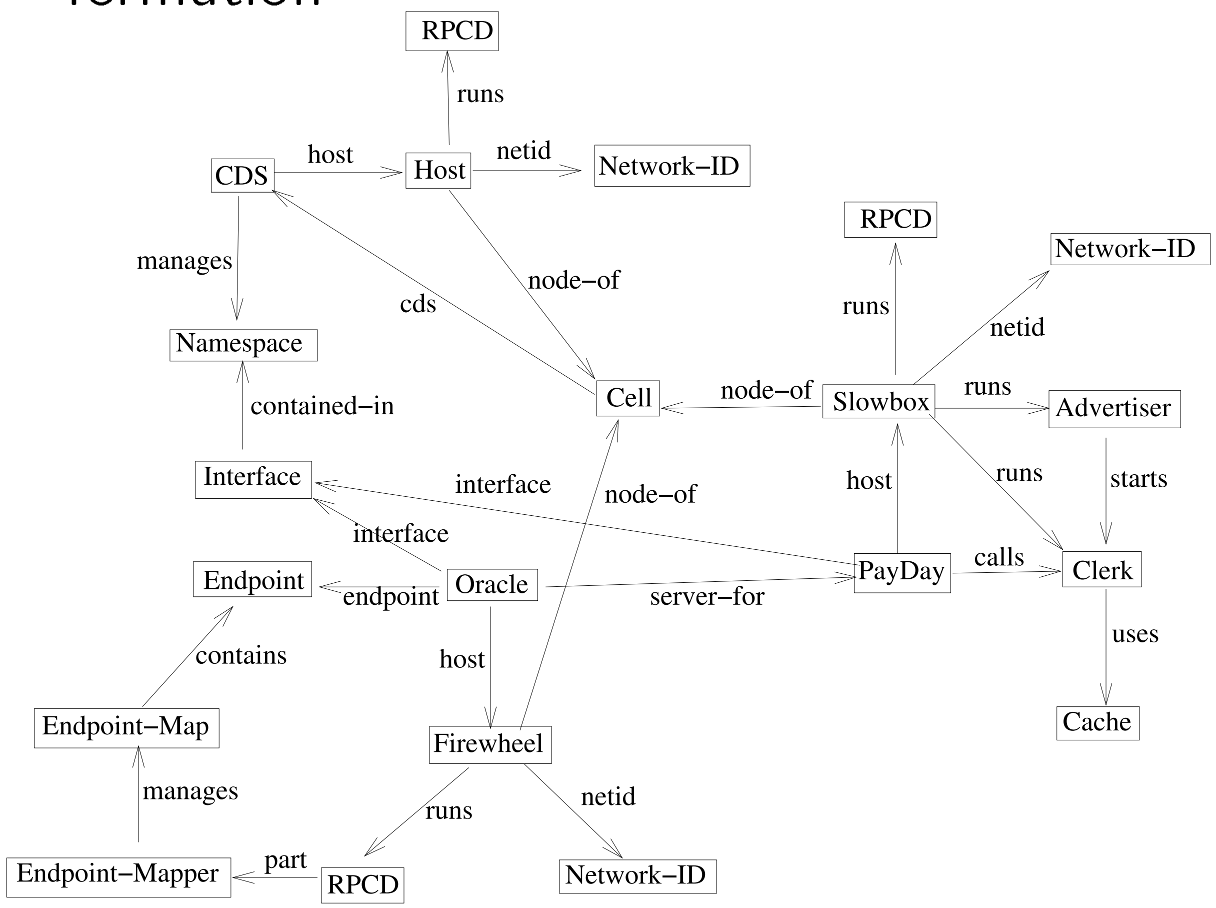
Peter Clark
Knowledge-Based Systems Lab
Univ. Texas at Austin

Our Group:

- **Last 10 years:** Research in building and using large-scale, multifunctional knowledge bases
- **Biology KB:** 20,000 concepts, 100,000 assertions. Used by tutoring applications.
- **Computing KB:** Smaller, task-neutral KB, used for help-desk applications.

What is a Knowledge Base?

- Not a text-base
- A machine-reasonable representation of information



Motivation

Currently, many knowledge-based systems:

- are task-specific
- often contain 'shallow' representations
- may contain similar information
- need to be redesigned from scratch as demands change

Problems

Short shelf-life: Effort to build systems is not amortized.

Redundant Effort: Knowledge common to multiple applications isn't shared

Difficult to maintain: Multiple versions of the same knowledge to keep up-to-date

Single-Task Knowledge Bases: Examples

Knowledge Base	Domain	Task
MYCIN	Medicine	Diagnosis
Dipmeter	Geology	Interpretation
Protos	Medicine	Diagnosis

Knowing the task simplifies:

- Knowledge representation
- Knowledge acquisition

The Space of Domain-Tasks

DOMAIN \ TASK	Diagnosis	Prediction	Tutoring	Synthesis	Assembly	...
Medicine						
Geology						
Computing						
...						

Traditional KB: One square

Multifunctional KB: One row

Two lessons learned from 70's & 80's:

1. Much domain knowledge shared among tasks
2. Difficult to integrate KBs:
   - Can't just union them
   - Requires a deliberate effort to build a more task-neutral KB

Project Goals

- Avoid 'start-from-scratch' paradigm by representing knowledge in a **general, task-neutral** form
- Amortize the effort required to build knowledge bases
- Achieve leverage by identifying and representing the general abstractions underlying domain-specific rules.

The Computing KB Project: Application Goals

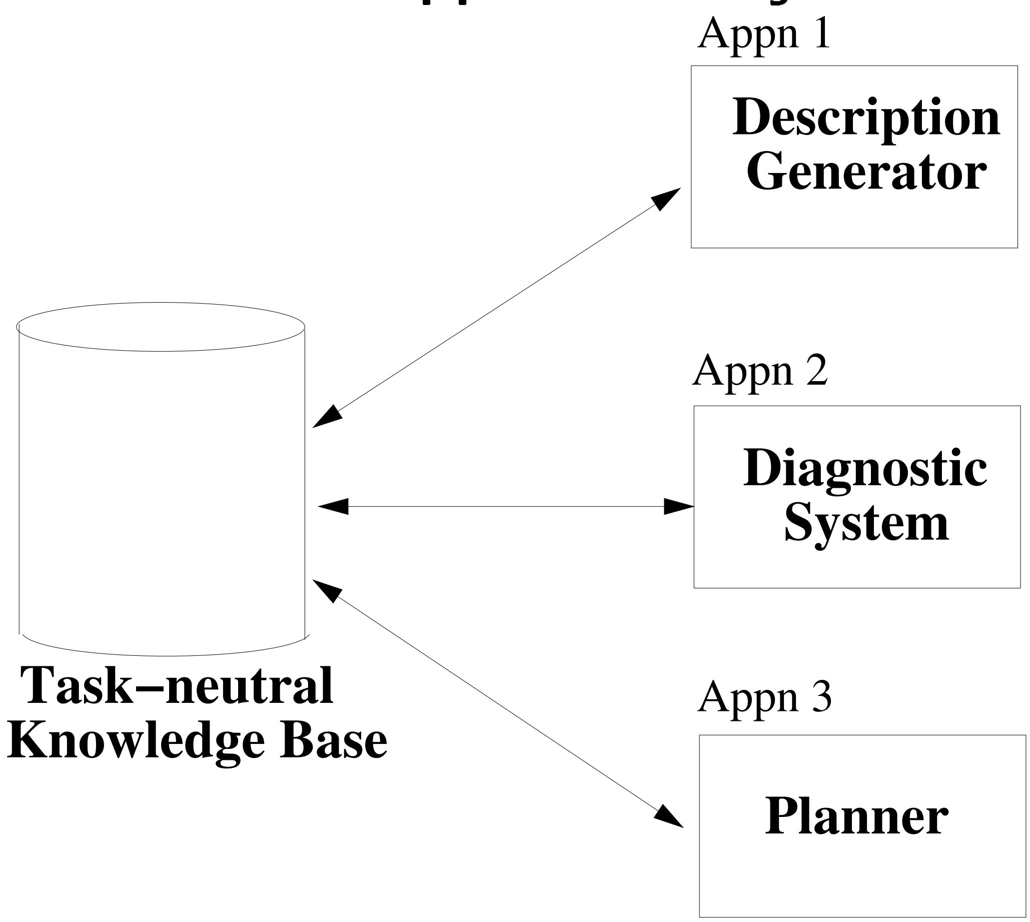
- Provide **automated help-desk services** to novice users.
- Three specific application tasks targeted:
  - **Description** ("What is...?")
  - **Diagnosis** ("Why did...?")
  - **Planning** ("How do I...?")
- Drive applications using a **shared knowledge-base** about computing (DCE).

The Applications

Use 'off-the-shelf' AI techniques to **infer** rather than recite, answers:

- $\rightarrow$ Answer's customized to user's situation.
- $\rightarrow$ Customize level of detail.

The Three Application Systems



Overlap in knowledge requirements of each application $\Rightarrow$ use single, shared KB.

The Knowledge Base: Some Issues

Representation language? Based on **conceptual graphs**

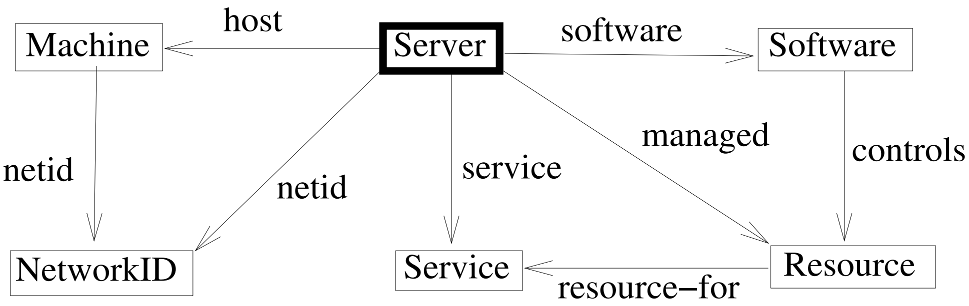
Access? using a SQL-like **query language**

Inference? Has built-in **inference engine**

Interaction? Applications interact with the KB via an **on-going dialog**.

A Representational Unit

(i) Conceptual Graph Notation



(ii) In FOL:

$$\forall S{:}server \;\; \exists M{:}machine \;\; host(S,M).$$
$$\forall S{:}server \;\; \exists V{:}service \;\; service(S,V).$$
$$\forall S{:}server,\; M{:}machine,\; N{:}netid$$
$$host(S,M) \wedge netid(M,NetId) \rightarrow netid(S,NetId).$$
$$\forall S{:}server,\; R{:}resource,\; V{:}service$$
$$managed(S,R) \wedge service(S,V) \rightarrow resource\text{-}for(R,V).$$

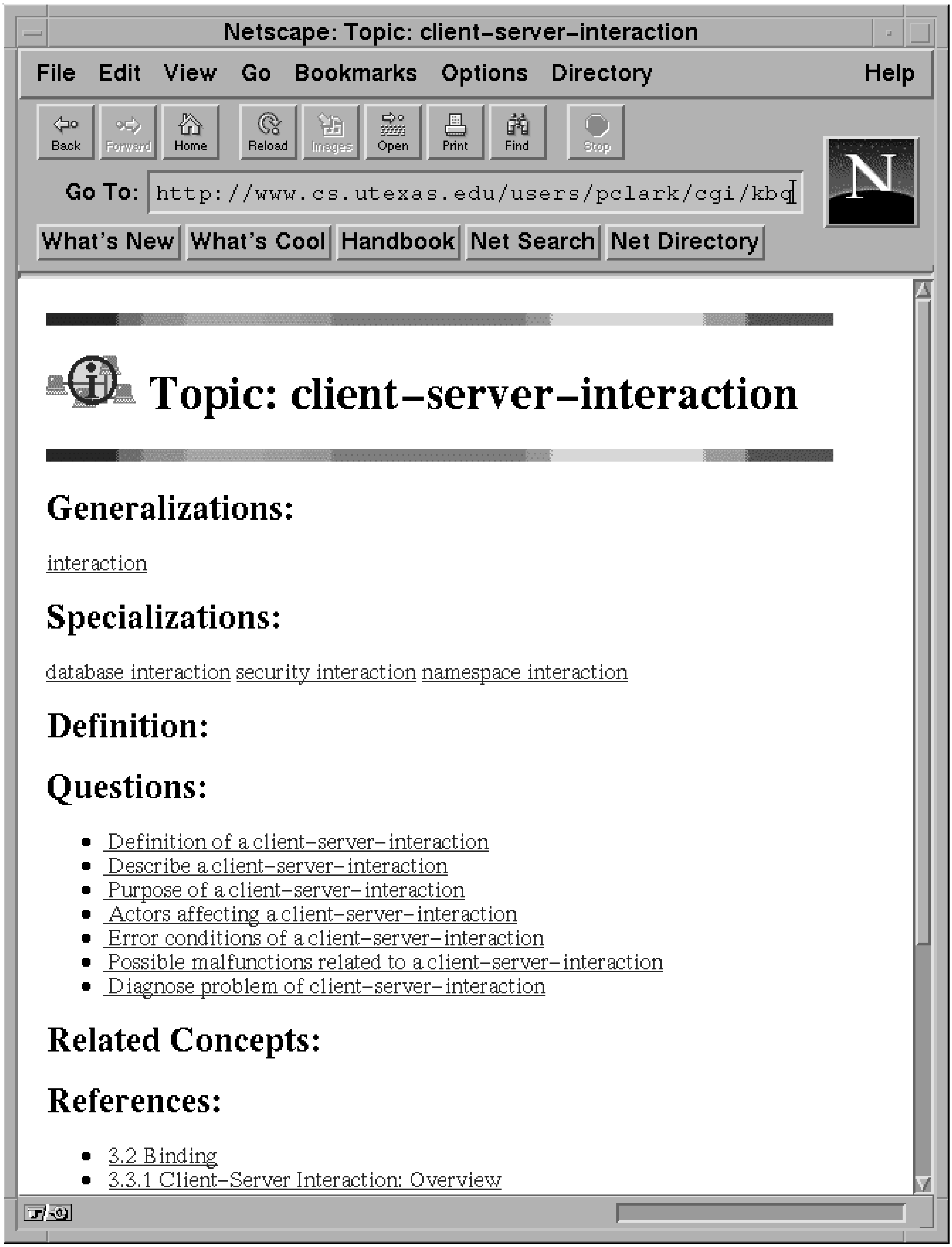
(iii) Internal Data Structure:

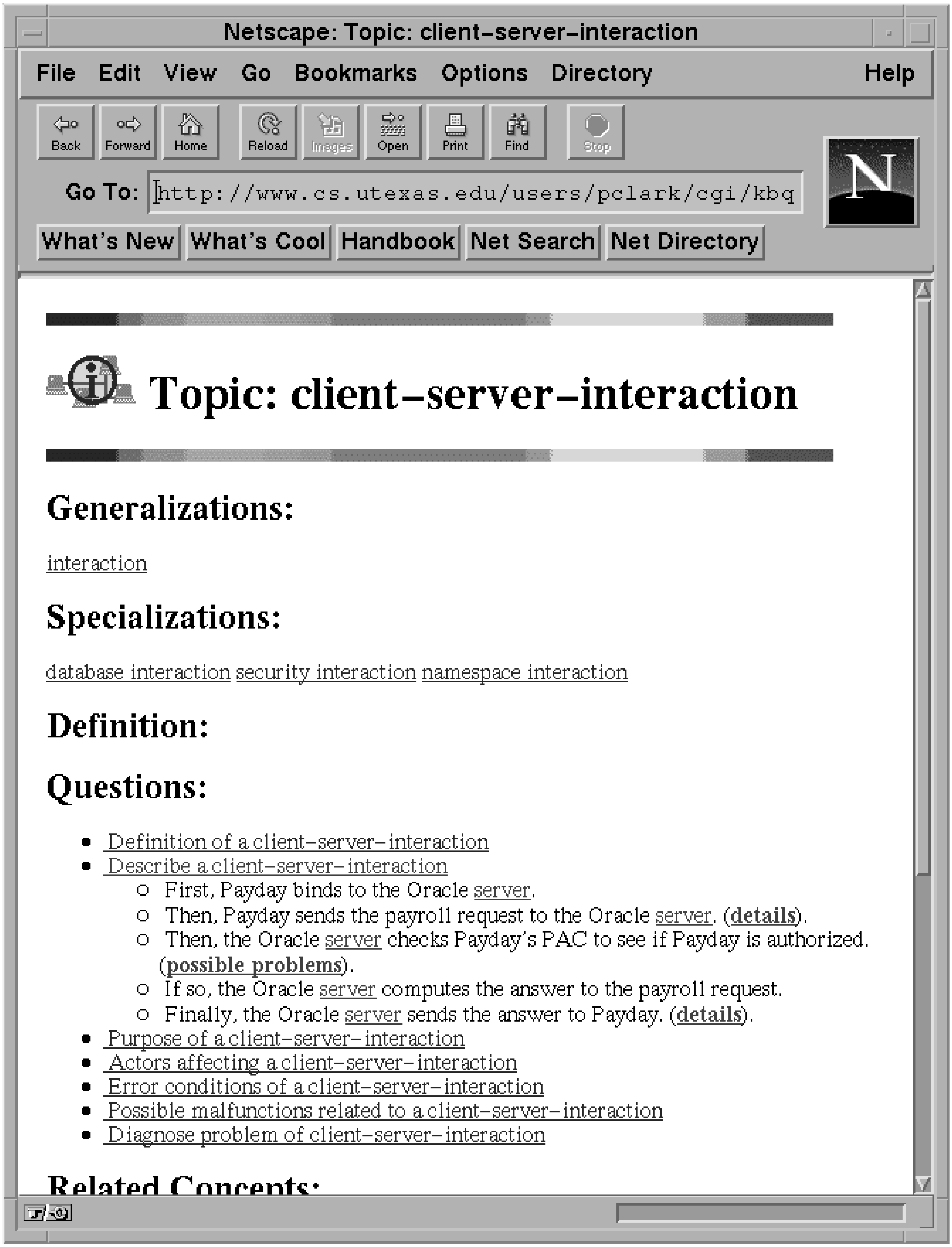
```
Server
------
  host: Machine
  service: Service
  network-id: (the network-id of (the host of Self))
  managed: Resource
           --------
           resource-for: (the service of Self)
  software: Software
            --------
            controls: (the managed of Self)
```

Building a KB from Components

Now I will:

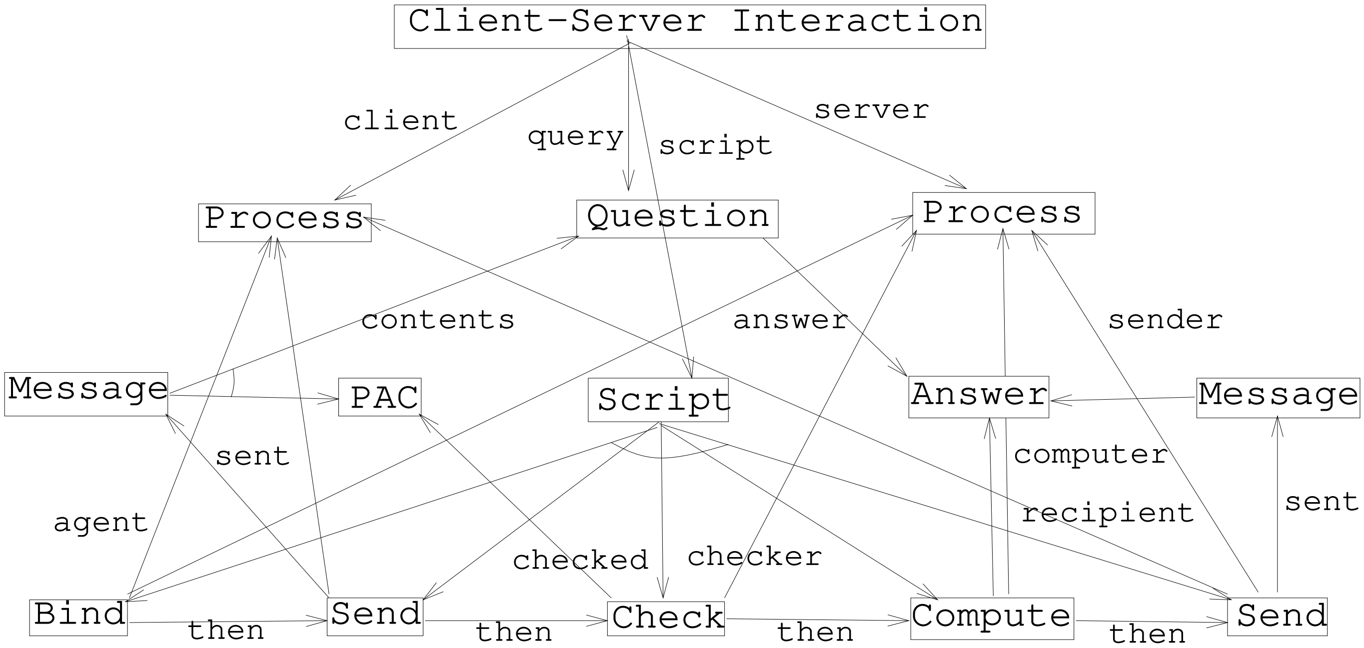
1. Illustrate an application answering a question
2. Show what the KB must provide for the appn to generate the answer
3. Show how this knowledge can be specified as a **composition** of components
4. Show how this composition can be performed at run-time


Netscape: Topic: client-server-interaction
File Edit View Go Bookmarks Options Directory Help
Back Forward Home Reload Images Open Print Find Stop
Go To: http://www.cs.utexas.edu/users/pclark/cgi/kbq
What's New What's Cool Handbook Net Search Net Directory
Topic: client-server-interaction
Generalizations:
interaction
Specializations:
database interaction security interaction namespace interaction
Definition:
Questions:
Definition of a client-server-interaction
Describe a client-server-interaction
Purpose of a client-server-interaction
Actors affecting a client-server-interaction
Error conditions of a client-server-interaction
Possible malfunctions related to a client-server-interaction
Diagnose problem of client-server-interaction
Related Concepts:
References:
3.2 Binding
3.3.1 Client-Server Interaction: Overview



Netscape: Topic: client-server-interaction
File Edit View Go Bookmarks Options Directory Help
Back Forward Home Reload Images Open Print Find Stop
Go To: http://www.cs.utexas.edu/users/pclark/cgi/kbq
What's New What's Cool Handbook Net Search Net Directory
Topic: client-server-interaction
Generalizations:
interaction
Specializations:
database interaction security interaction namespace interaction
Definition:
Questions:
Definition of a client-server-interaction
Describe a client-server-interaction
First, Payday binds to the Oracle server.
Then, Payday sends the payroll request to the Oracle server. (details).
Then, the Oracle server checks Payday's PAC to see if Payday is authorized. (possible problems).
If so, the Oracle server computes the answer to the payroll request.
Finally, the Oracle server sends the answer to Payday. (details).
Purpose of a client-server-interaction
Actors affecting a client-server-interaction
Error conditions of a client-server-interaction
Possible malfunctions related to a client-server-interaction
Diagnose problem of client-server-interaction
Related Concepts:

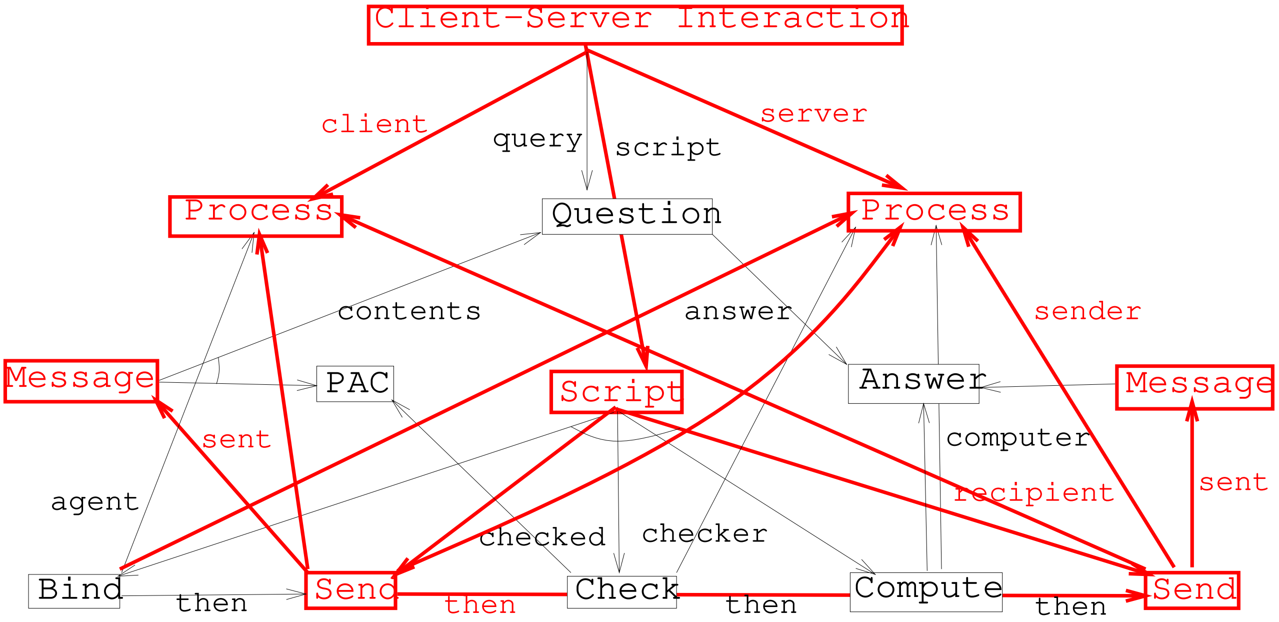
Underlying Structure Required to Answer the Question:

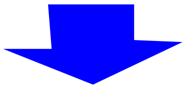


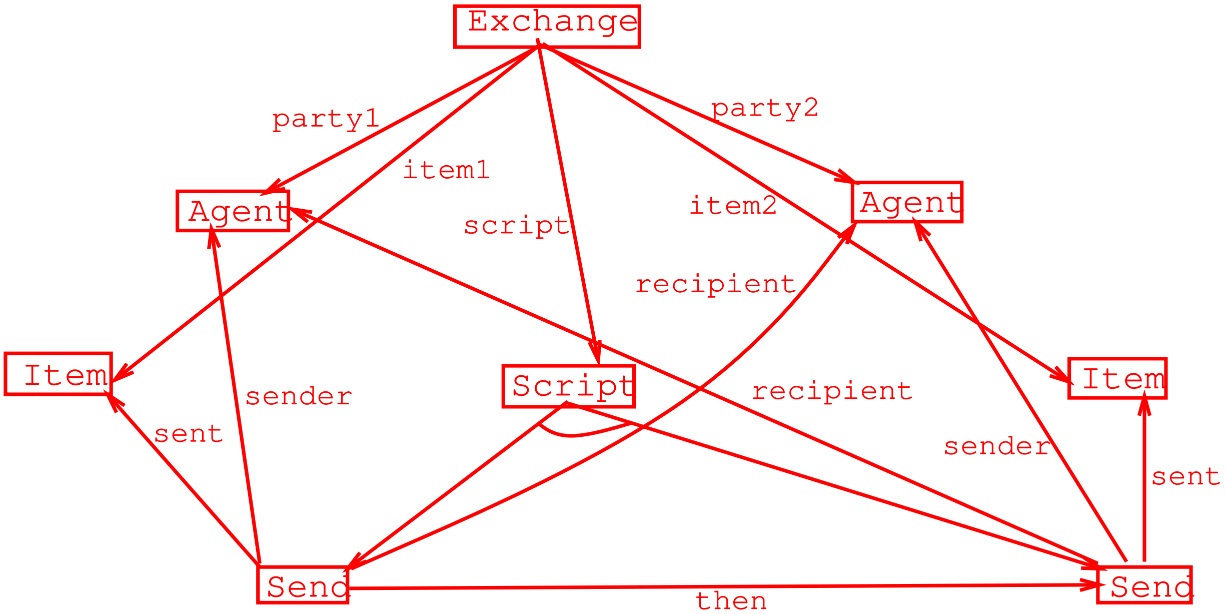
Building a KB From Components

- **Happy:** Given such structures, can answer:
  - Description questions
  - Diagnostic questions
  - Planning questions
- **Unhappy**:
  - Representations laborious to build
  - Only answer anticipated questions
  - DCE-specific
- **But:** Patterns exist in these structures
- $\Rightarrow$ **Goal:** Build representations from a library of components
- **Component** = intermediate-level abstraction/cliche

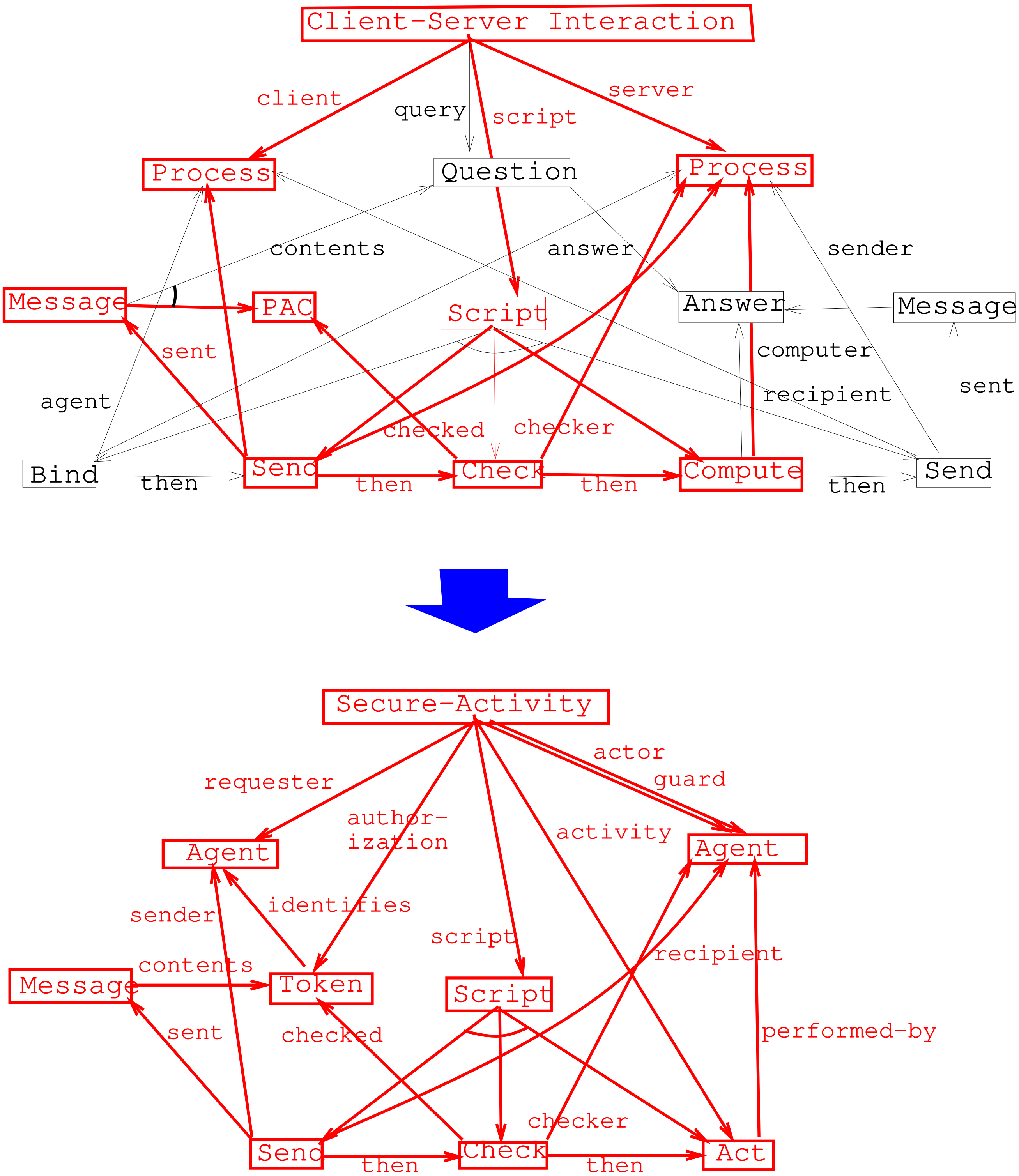
‘Exchange’ component of ‘Client–Server Interaction’







‘Security’ component of ‘Client–Server Interaction’



Client−Server Interaction specified as a Composition

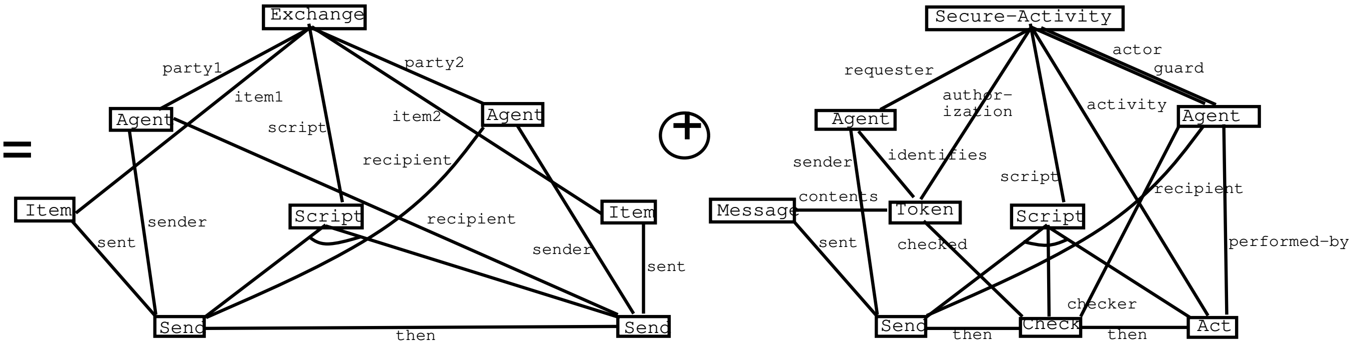
Client−Server Interaction =

Exchange
party1 = the client
party2 = the server
item1 = a message with contents = the query
item2 = a message with contents = the answer of the query

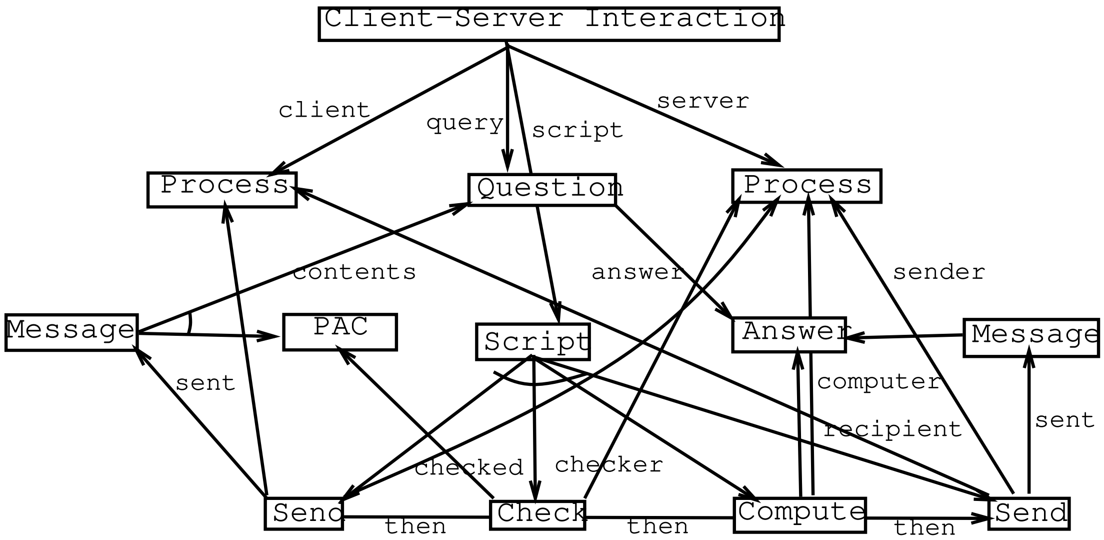
⊕

Secure−Activity
requester = the client
actor = the server
guard = the server
authorization = a PAC
activity = a compute of the answer of the query

=

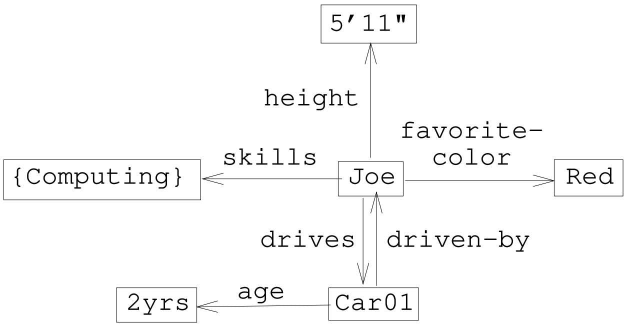


=

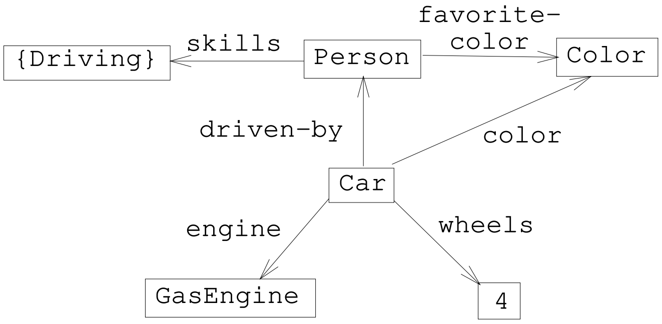


Graph Unification

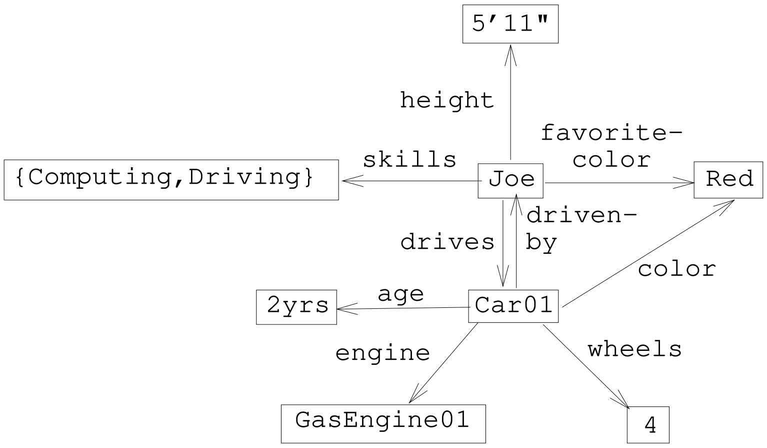
- Merges two graphs together
- Output graph contains all the info implied by each input graph.



&



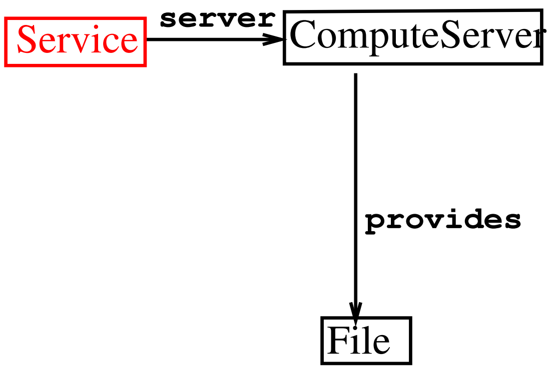
=



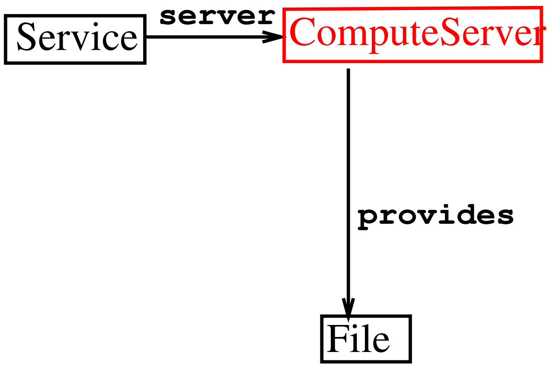
Building the Graph Dynamically to Answer Questions

"the supporting h/w of the
server of the file-service?"

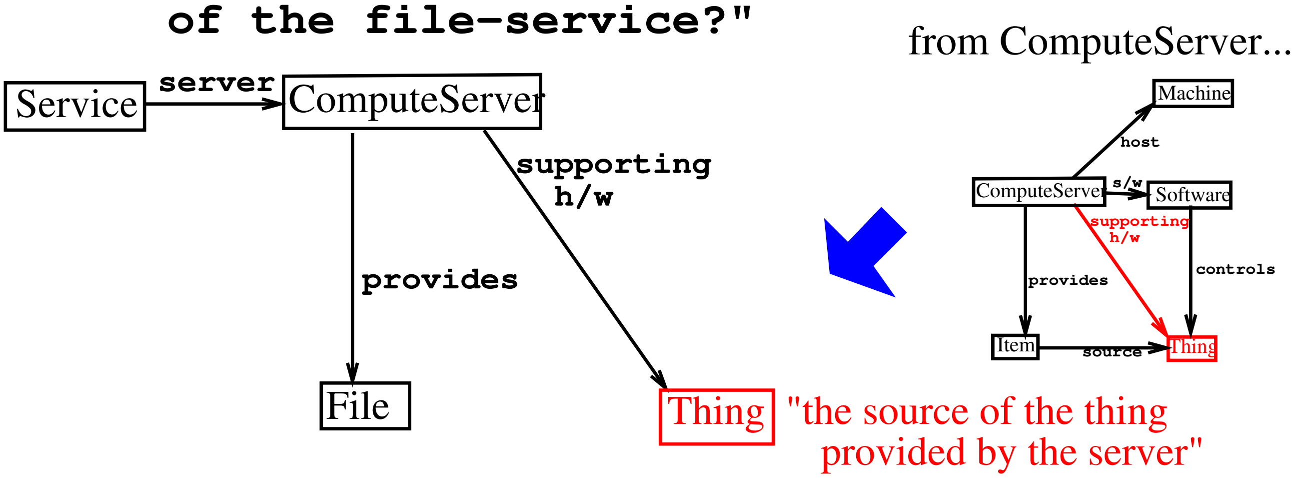
1. "the file-service?"



2. "the server of the file-service?"

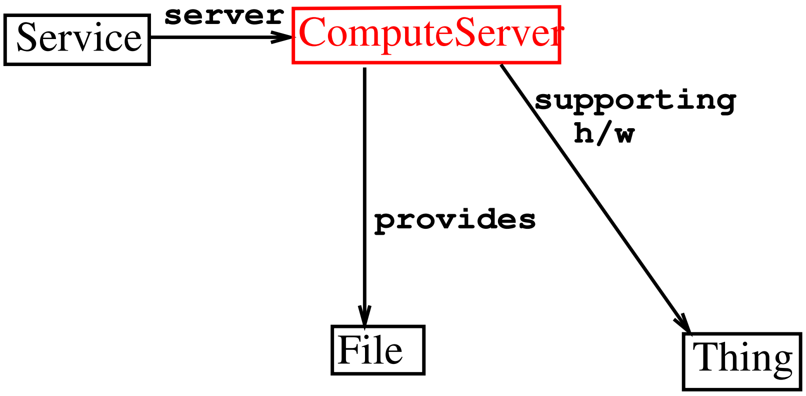


3. "the supporting h/w of the server
of the file-service?"

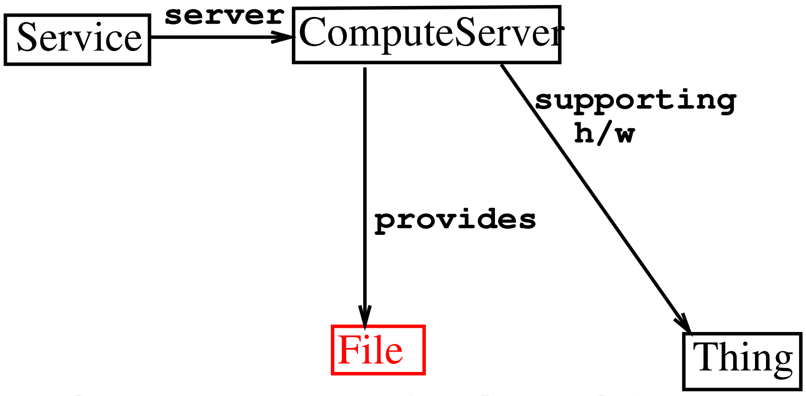


"the source of the thing provided by the server?"

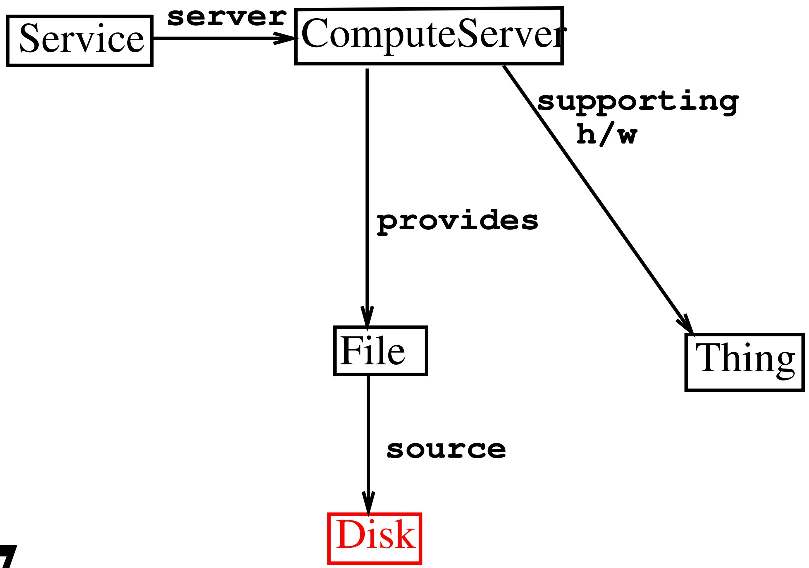
4. "the server?"

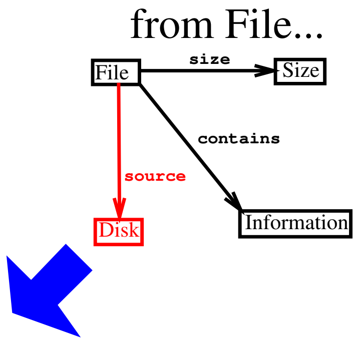


5. "the thing provided by the server?"



6. "the source of the thing provided by server?"





7. "a disk!"

Examples of Components:

address	identifier
agent	lock
communication	machine
conversion	representation
container	security
destruction	service
exchange	software
information	token

Some Compositions:

```
database = secure lockable container
file = representation of information
server = software agent
machine = software agent's host
disk = file collection
delete(file) = destroy
binding = server address
ticket = time-limited token of identity
```

Summary

- **Goal:** Amortize effort to build KBSs by having a **centralized, task-neutral** knowledge base.
- **Problems:**
  - KBs are difficult to build
  - Danger of KB becoming a large, unwieldy mass of highly specific knowledge
- **Approach:** Build KB from a library of **general, reusable components**

Related Work

- ARPA **Knowledge Sharing Effort** (KIF, KQML, Ontolingua)
- **ARPI** Planning Initiative (KRSL)
- **Frames** & derivatives (KRL,Loom,Algernon)
- Cliches, MOPS
- **Conceptual Graphs** (Peirce)
- **Cyc** (Microtheories)
- **Compositional Modeling** (CML,Xerox)
- **s/w engineering** (Design Patterns, Component-based Architectures)
- **Also:** Work on reusable problem-solving methods (Protege, KADS, Generic Tasks)

Ongoing and Future Work

- Completion of component library
- ‘Closure’ of DCE Knowledge Base
- Re-engineer Biology KB as compositions