# Multi Agenten Systeme
## VU SS 00, TU Wien

Teil 1 (Kapitel 1–4) basiert auf

**Multi-Agent Systems** (Gerhard Weiss), MIT Press, June 1999.

Es werden allgemeine Techniken und Methoden dargestellt (BDI-, Layered-, Logic based Architekturen, Decision Making, Kommunikation/Interaktion, Kontrakt Netze, Coalition Formation).

Teil 2 (Kapitel 5–9) basiert auf

**Heterogenous Active Agents** (Subrahmanian, Bonatti, Dix, Eiter, Kraus, Özcan and Ross), MIT Press, May 2000.

Hier wird ein spezifischer Ansatz vorgestellt, der formale Methoden aus dem logischen Programmieren benutzt, aber nicht auf PROLOG aufsetzt (Code Call Mechanismus, Aktionen, Agenten Zyklus, Status Menge, Semantiken, Erweiterungen um Beliefs, Implementierbarkeit).

# Übersicht

# Chapter 5. *IMPACT* Architecture

## Overview

## 5.1 Three Szenarios

## 5.2 Agent Architecture

## 5.3 Server Architecture

## 5.4 Service Description Language

# 5 *IMPACT* Architecture

## 5.1 Three Szenarios

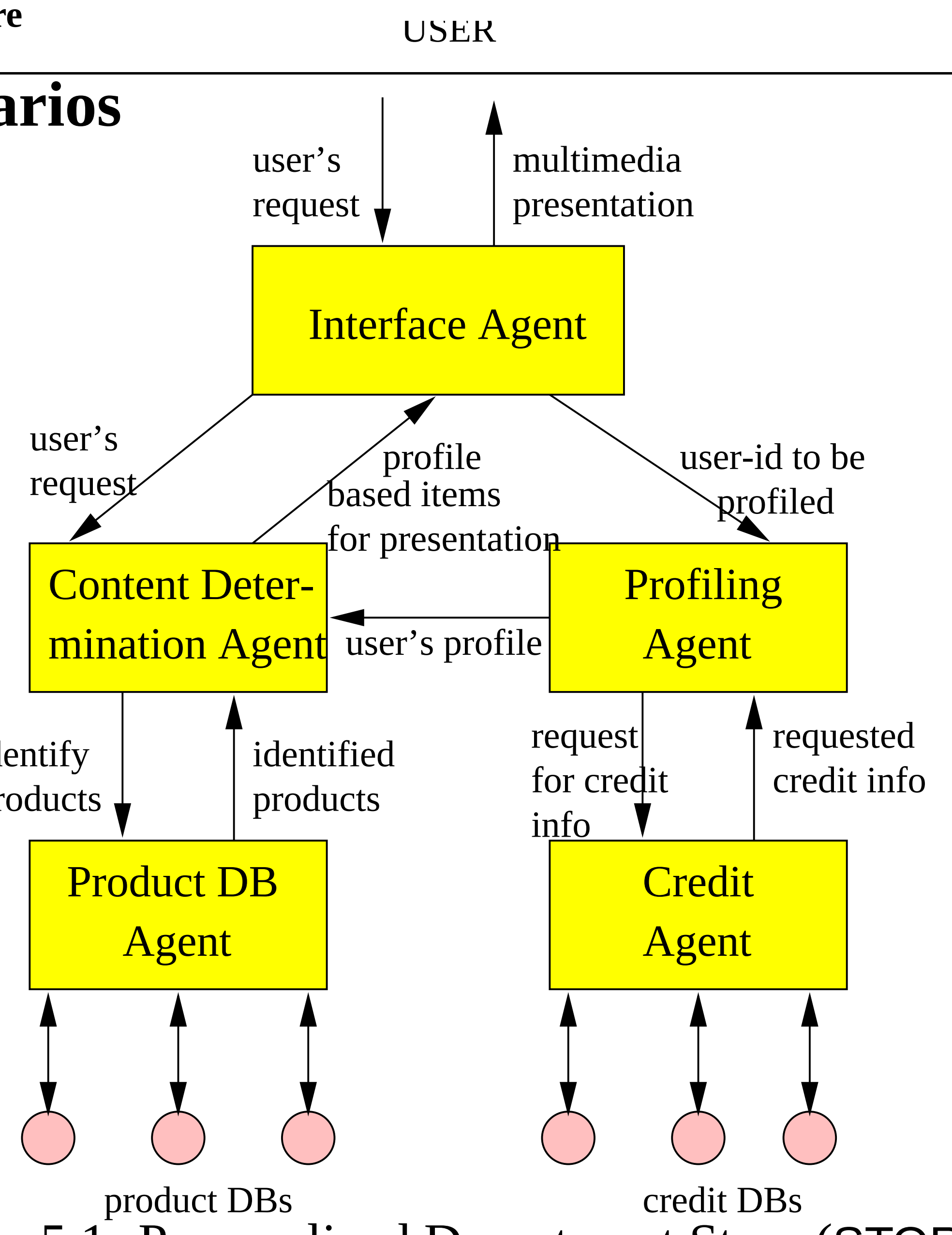



Figure 5.1: Personalized Department Store (STORE)

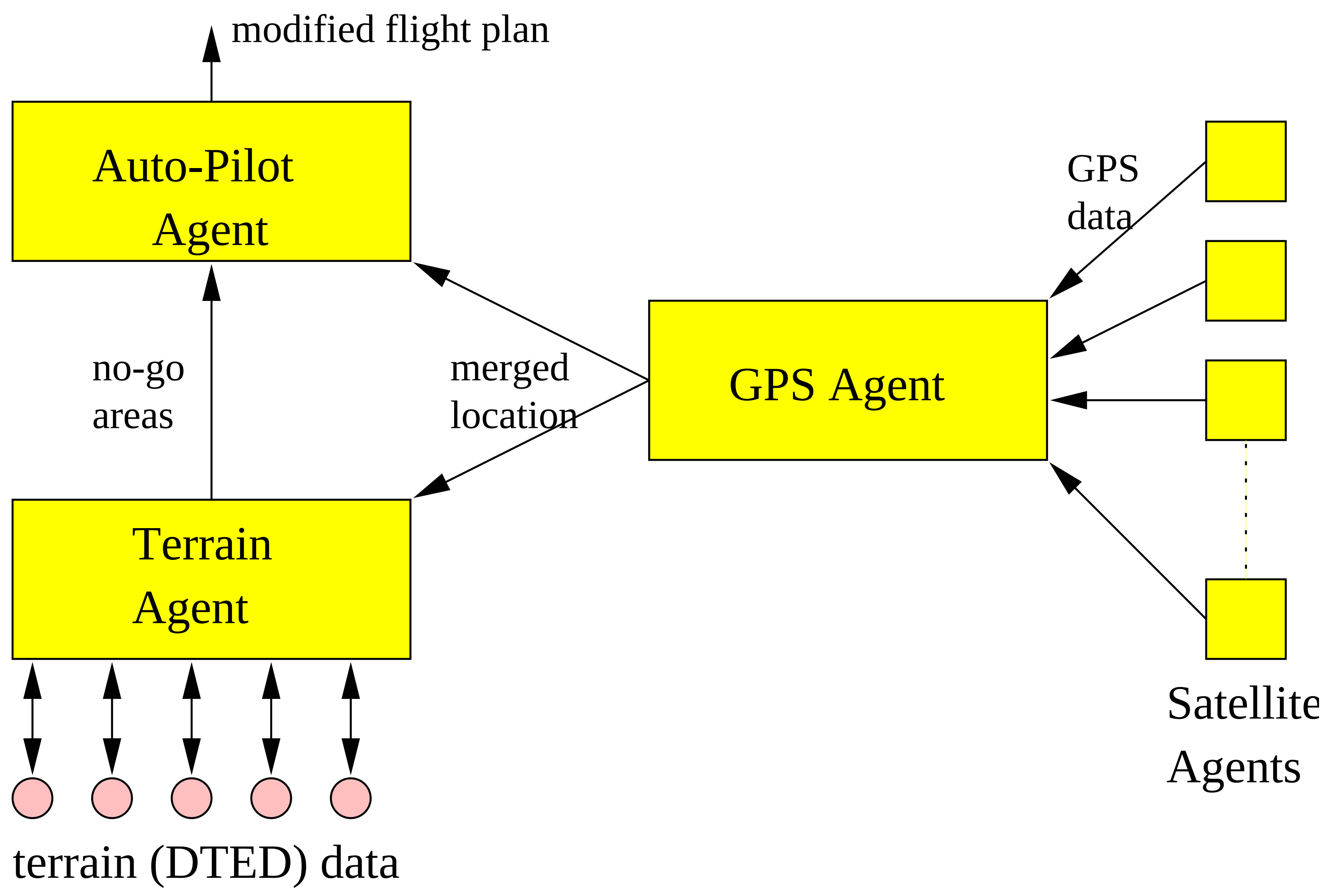



Figure 5.2: Interactions between Agents in CFIT Example

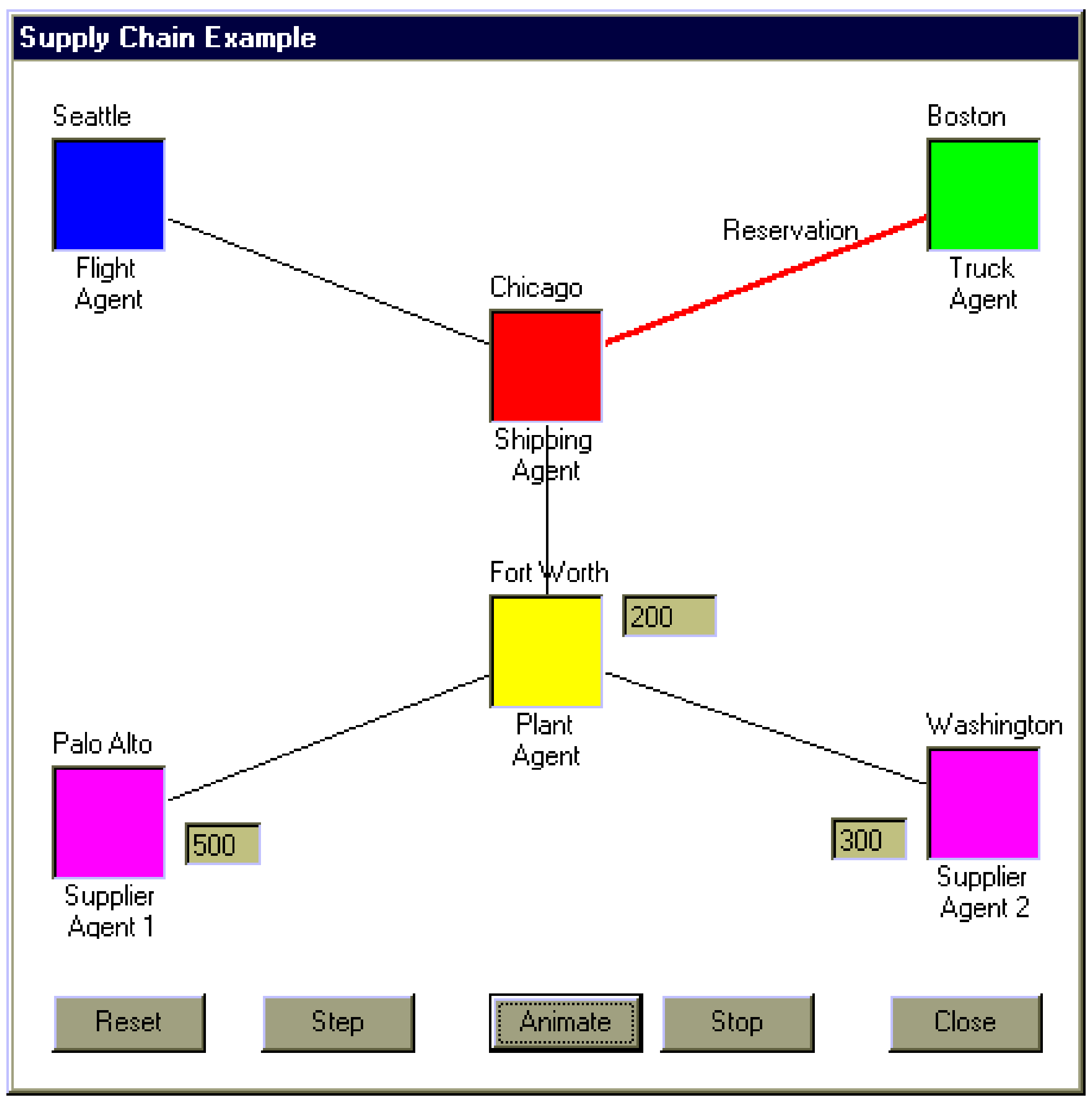



Figure 5.3: Agents in CHAIN Example

## 5.2 Agent Architecture

Four main categories:

1. In the first category, each agent has an associated **“transducer”** that converts all incoming messages and requests into a form that is intelligible to the agent. In general, in an $n$-agent system, we may need $O(n^2)$ transducers, which is clearly not desirable.

2. The second approach is based on **wrappers** which **“inject code into a program to allow it to communicate”** (Genesereth and Ketchpel 1994, p. 51). This idea is based on the principle that each agent has an associated body of code that is expressed in a common language used by other agents (or is expressed in one of a very small number of such languages).

3. The third approach described in (Genesereth and Ketchpel 1994) is to **completely rewrite the code** implementing an agent, which is obviously a very expensive alternative.

4. Last but not least, there is the **mediation approach** proposed by Wiederhold (1993), which assumes that all agents will communicate with a mediator which in turn may send messages to other agents. The mediation approach has been extensively studied (Arens, Chee, Hsu, and Knoblock 1993; Brink, Marcus, and Subrahmanian 1995; Chawathe, S., et al. 1994; Bayardo, R., et al. 1997).

   **Here is the problem:** Suppose all communications in the CFIT example had to go through such a mediator. Then if the mediator malfunctions or "goes down," the system as a whole is liable to collapse, leaving the plane in a precarious position.

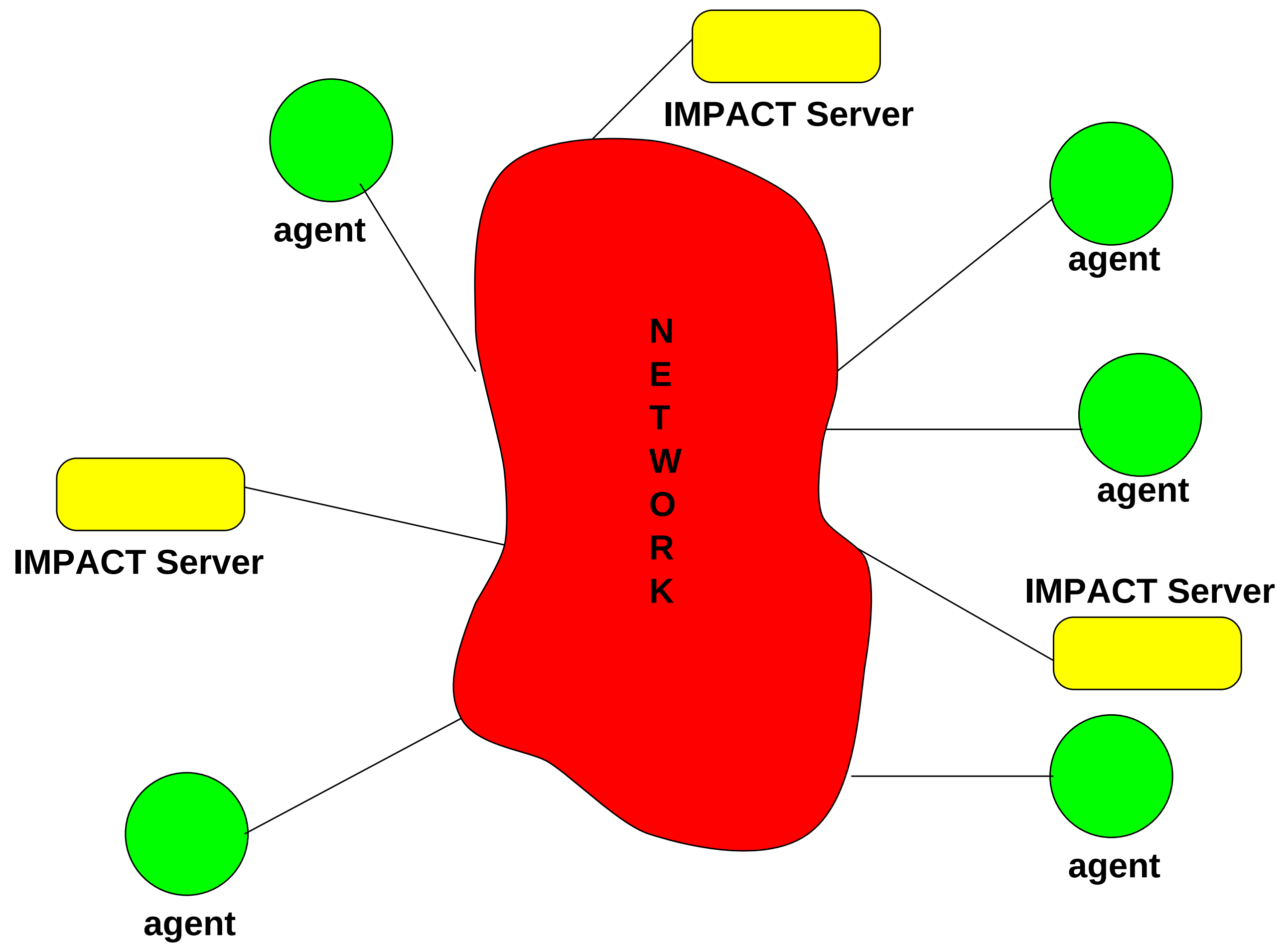



Figure 5.4: Overall *IMPACT* Architecture

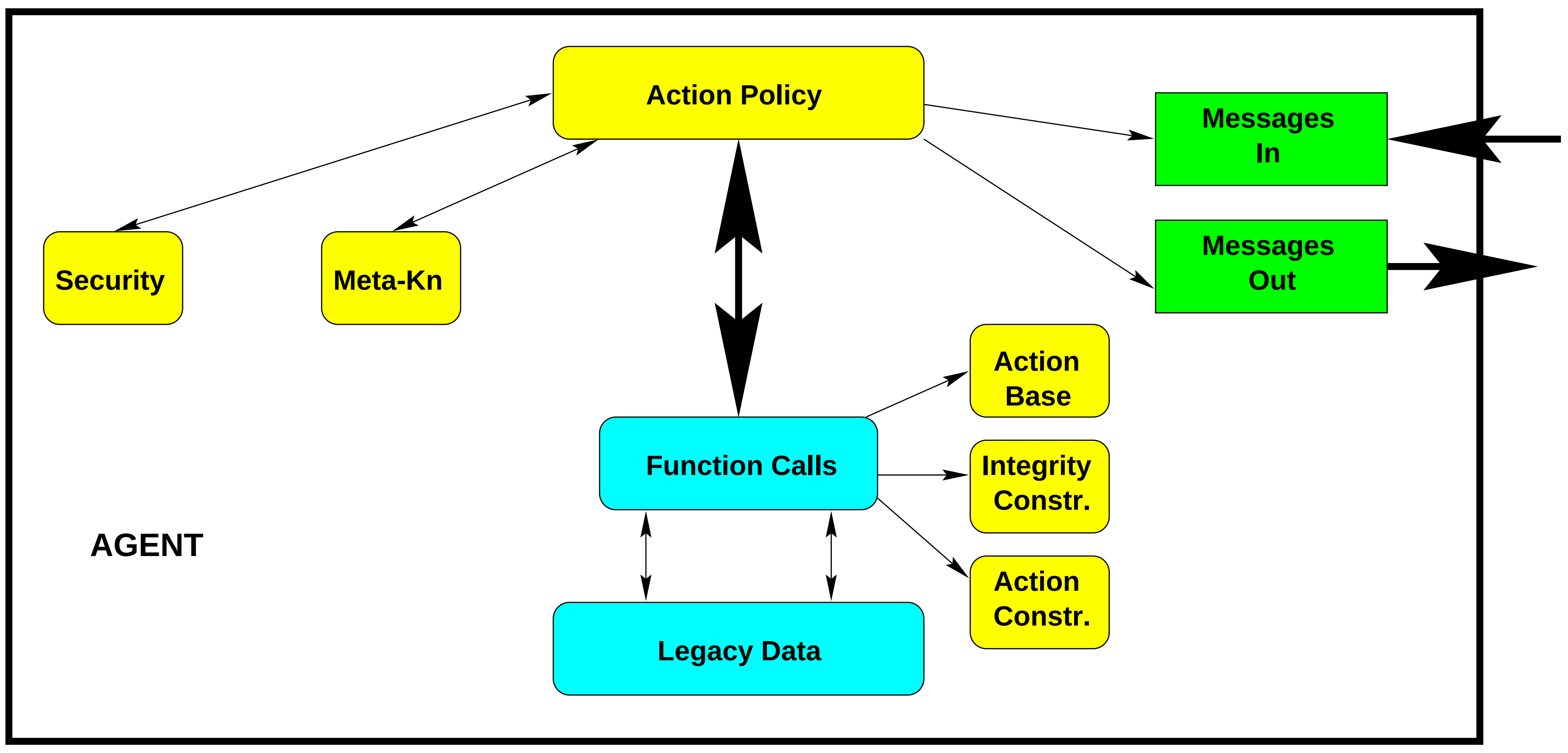



Figure 5.5: Basic Architecture of *IMPACT* Agents

## 5.3 Server Architecture

An *IMPACT* Server is actually a collection of the following servers:

**Registration Server:** This server is mainly used by the creator of an agent to specify the services provided by it and who may use those services.

**Yellow Pages Server:** This server processes requests from agents to identify other agents that provide a desired service.

**Thesaurus Server:** This server receives requests when new agent services are being registered as well as when the yellow pages server is searching for agents providing a service.

**Type Server:** This server maintains a set of class hierarchies containing information about different data types used by different agents, and the inclusion relationship(s) between them.

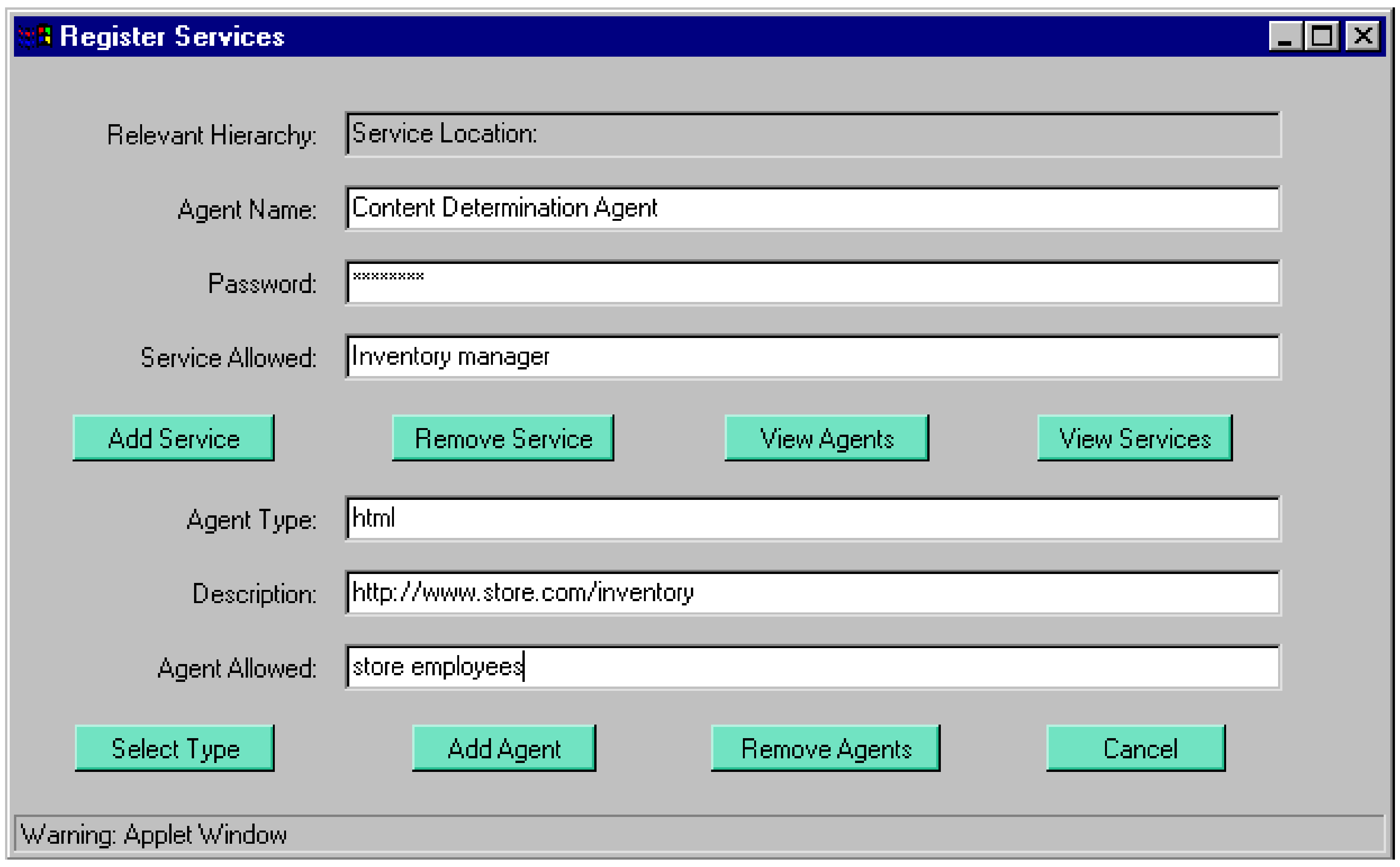



Figure 5.6: Agent/Service Registration Screen Dump

The user needs to specify the services of his agent. This is done in SDL (Service Description Language).

**Definition 5.1 (Verbs, Nouns, nt(Nouns))**
*Suppose* Verbs *is a set of verbs in English, and* Nouns *is a set of nouns in English.*

- *A* noun term *is either a noun or an expression of the form* $n_1(n_2)$ *where* $n_1, n_2$ *are both nouns.*
- nt(Nouns) *denotes the set of all syntactically valid noun terms generated by the set* Nouns*.*

**Definition 5.2 (Service Name)**
*If* $v \in$ Verbs *and* $nt \in$ nt*, then* $v\!:\!nt$ *is called a* service name.

| AGENT | SERVICES |
|---|---|
| credit | *provide : information(credit)*<br>*provide : address* |
| profiling | *provide : user-profile*<br>*classify : user* |
| productDB | *provide : description(product)*<br>*identify : product* |
| contentDetermin | *prepare : presentation(product)*<br>*determine : advertisement*<br>*identify : items* |
| saleNotification | *identify : user-profile*<br>*determine : items* |

Table 5.1: Service List for the STORE example

| AGENT | SERVICE |
|---|---|
| autoPilot | *maintain : course*<br>*adjust : course*<br>*return : control*<br>*create : plan(flight)* |
| satellite | *broadcast : data(GPS)* |
| gps | *collect : data(GPS)*<br>*merge : data(GPS)*<br>*create : information(GPS)* |
| terrain | *generate : map(terrain)*<br>*determine : area(no-go)* |

Table 5.2: Service List for the CFIT example

| AGENT | SERVICE |
|---|---|
| plant | *monitor : inventory* |
| | *determine : amount(part)* |
| | *order : part* |
| | *notify : supplier* |
| supplier | *monitor : available-stock* |
| | *update : stock* |
| | *find : airplane* |
| | *prepare : schedule(shipping)* |
| truck | *provide : schedule(truck)* |
| | *manage : freight* |
| | *ship : freight* |

Table 5.3: Service List for the CHAIN example

What if one agent **a** seeks another one offering a service $q_s$?

We need to match $q_s$ with other services in the yellow pages.

An agent looks for an agent offering the service *generate : map(ground)*.

**Answer:** CFIT **terrain** agent: *ground* and *terrain* are synonymous.

Suppose $\Sigma$ is any set of English words, such that either all words in $\Sigma$ are verbs, or all words in $\Sigma$ are noun-terms. Furthermore, suppose $\sim$ is an arbitrary equivalence relation on $\Sigma$.

**Definition 5.3 ($\Sigma$-node)**
*A* $\Sigma$-node *is any subset* $N \subseteq \Sigma$ *that is closed under* $\sim$, *i.e.*

1. $x \in N \,\&\, y \in \Sigma \,\&\, y \sim x \Rightarrow y \in N$.
2. $x, y \in N \Rightarrow x \sim y$.

*In other words,* $\Sigma$-nodes *are equivalence classes of* $\Sigma$*.*

An agent looks for an agent offering the service *generate* : *map(area)*.

**Answer:** CFIT **terrain** agent: *area* can be specialized to *terrain*.

**Definition 5.4 (Σ-Hierarchy)**
*A Σ-Hierarchy is a weighted, directed acyclic graph* $\mathcal{SH} =_{def} (T, E, \wp)$ *such that:*

1. $T$ *is set of nonempty Σ-nodes;*
2. *If* $t_1$ *and* $t_2$ *are different Σ-nodes in* $T$*, then* $t_1$ *and* $t_2$ *are disjoint;*
3. $\wp$ *is a mapping from* $E$ *to* $\mathbb{Z}^+$ *indicating a positive distance between two neighboring vertices.*[a]

[a] We do not require $\wp$ to satisfy any metric axioms at this point in time.

## Verb Hierarchy

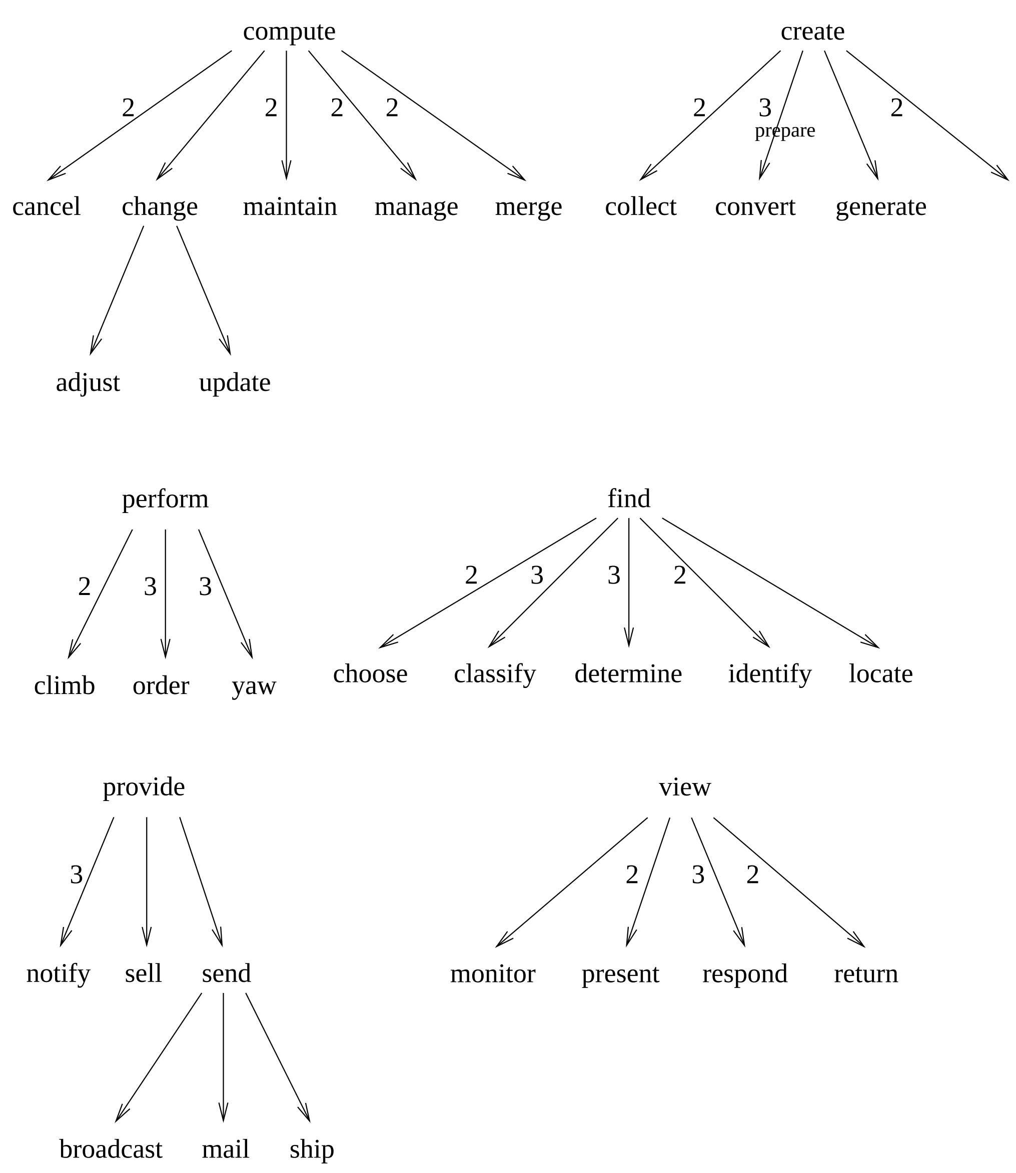



Figure 5.7: Verb Hierarchy (Missing Labels = 1)

## Noun Hierarchy

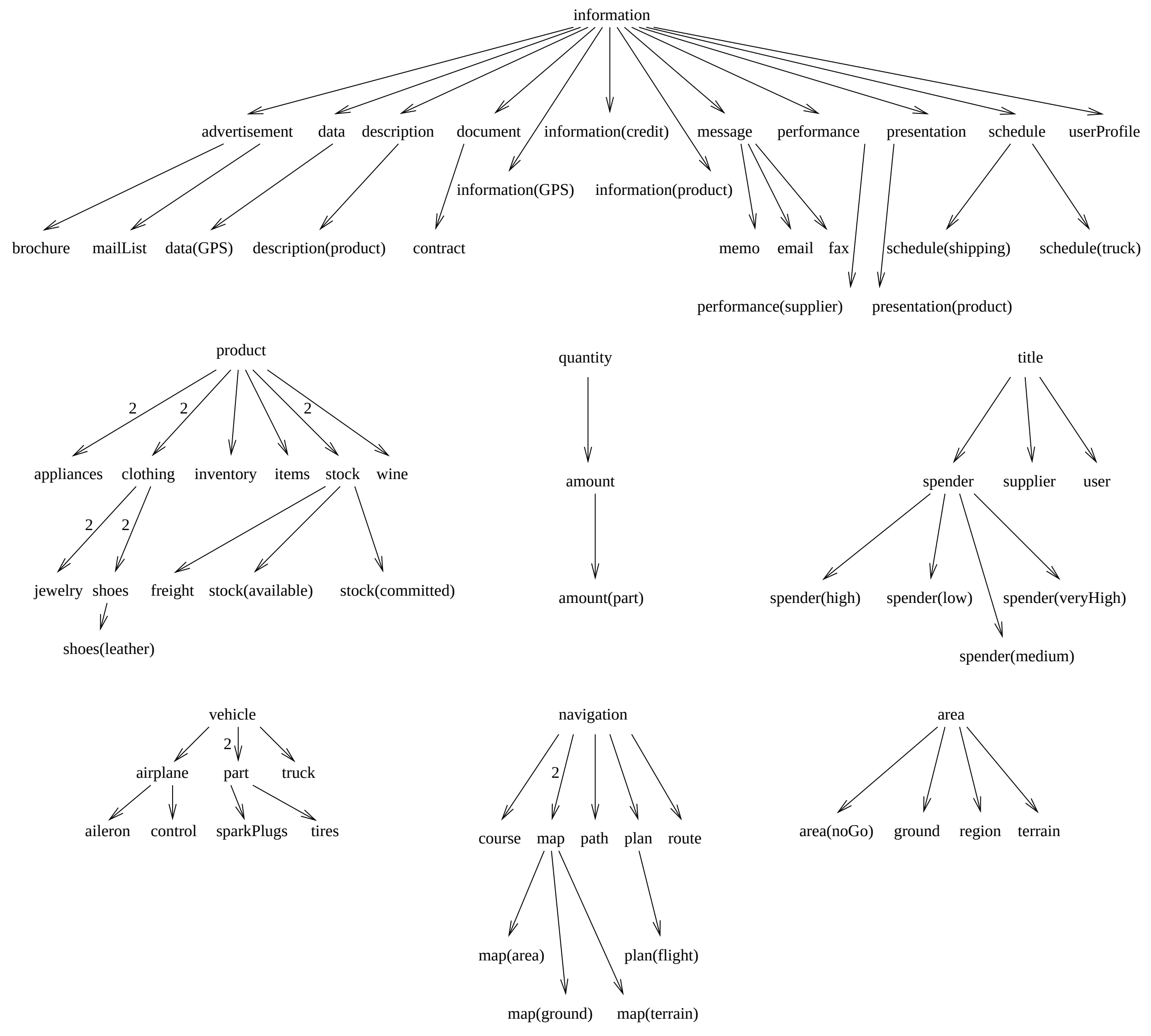



Figure 5.8: Noun-term Hierarchy

### 5.3.1 Distances

**Definition 5.5 (Distance between two terms)**

*Given a* $\Sigma$*-Hierarchy* $\mathcal{SH} =_{def} (T, E, \wp)$*, the distance between two terms,* $w_1, w_2 \in T$*, is defined as follows:*

$$\mathbf{d}_{\mathcal{SH}}(w_1, w_2) =_{def} \begin{cases} 0, & \text{if some } t \in T \text{ exists such that } w_1, w_2 \in t; \\ cost(p_{min}), & \text{if there is an undirected path in } \mathcal{SH} \text{ between} \\ & w_1, w_2 \text{ and } p_{min} \text{ is the least cost such path;} \\ \infty, & \text{otherwise.} \end{cases}$$

It is easy to see that given any $\Sigma$-hierarchy, $\mathcal{SH} =_{def} (T, E, \wp)$, the distance function, $\mathbf{d}_{\mathcal{SH}}$ induced by it is well defined and satisfies the triangle inequality.

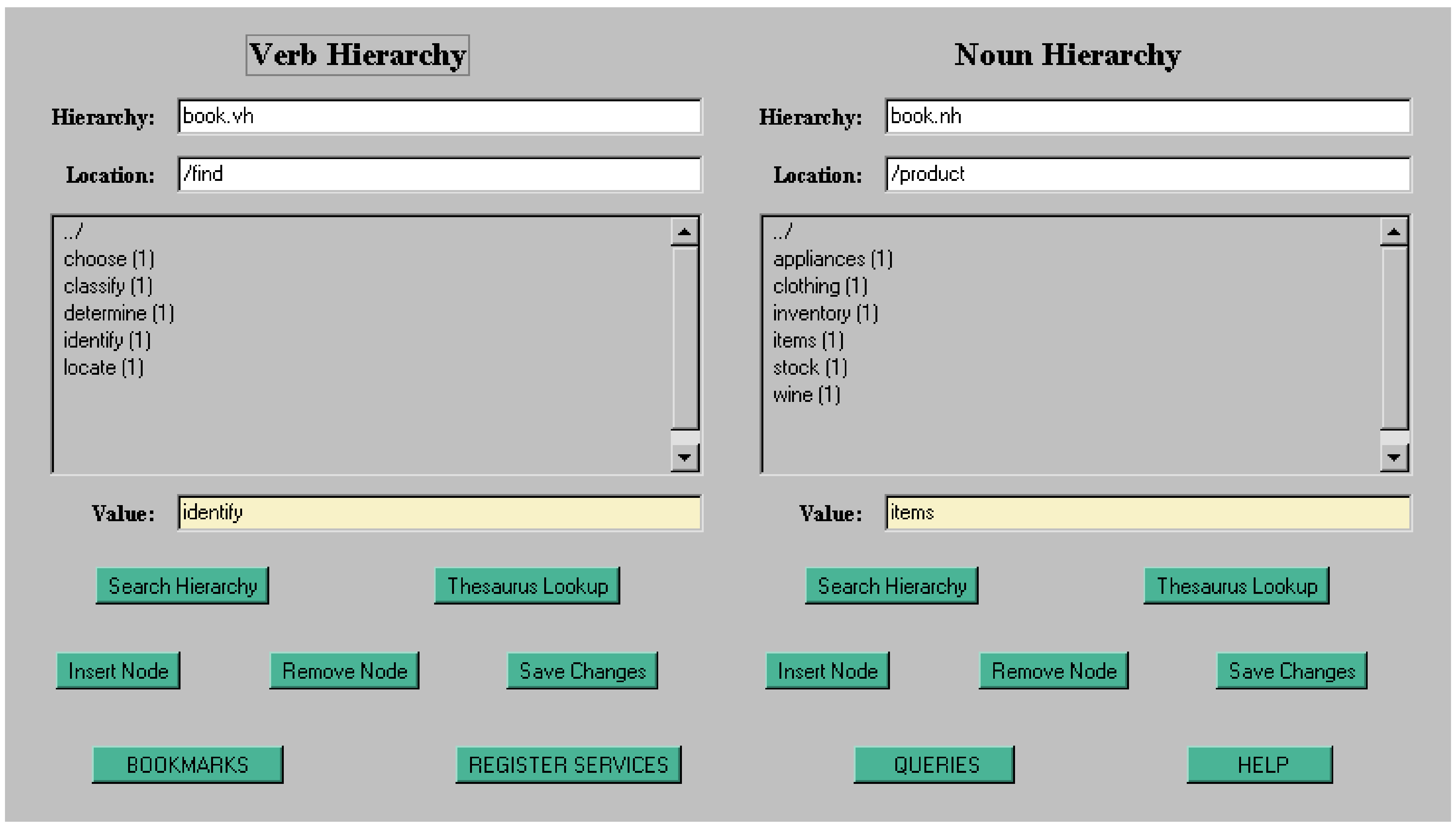



Figure 5.9: Hierarchy Browsing Screen Dump

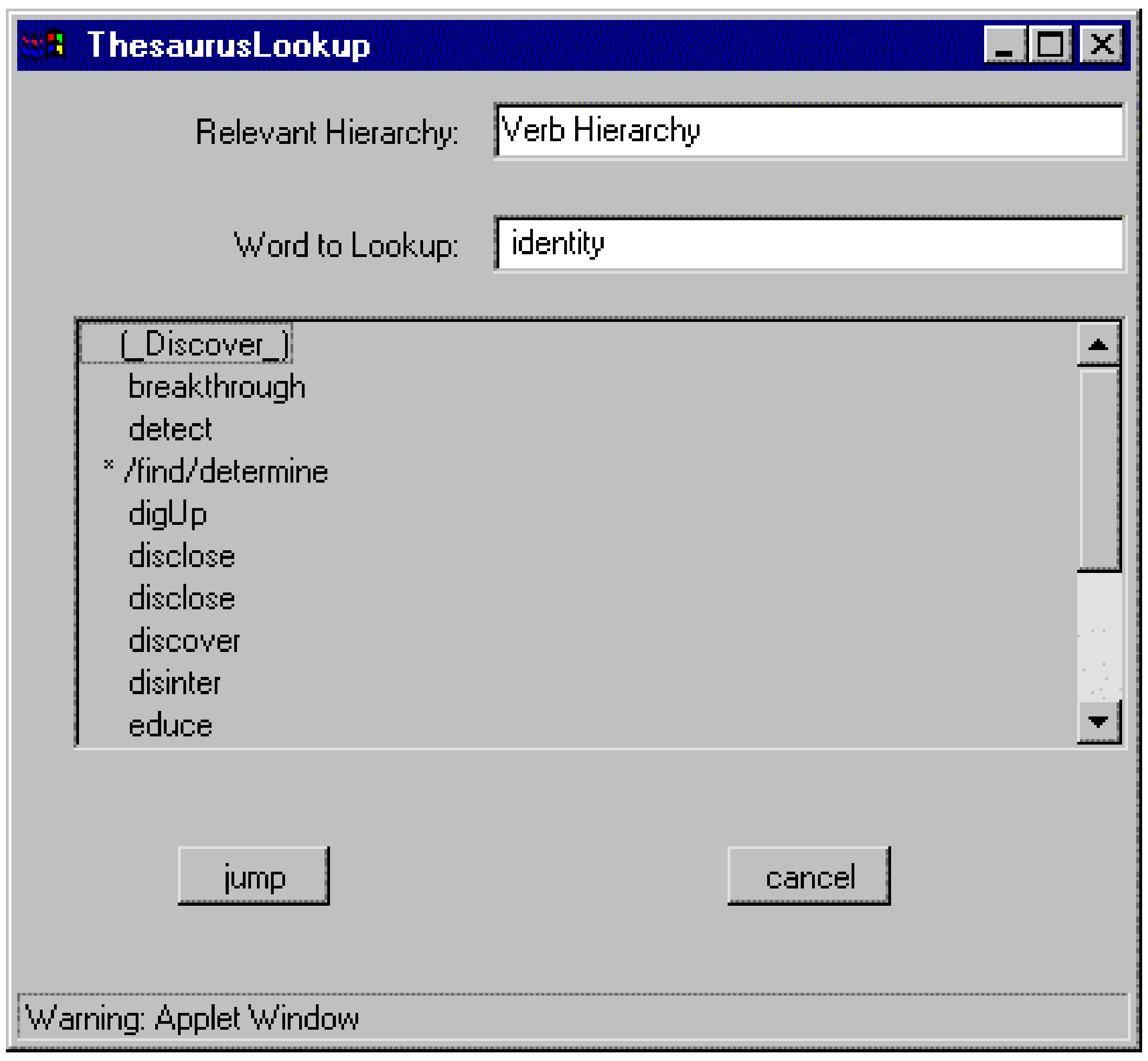



Figure 5.10: Thesaurus Screen Dump

## 5.4 Service Description Language

### 5.4.1 Definition of SDL

**Service Name:** This is a *verb* : *noun(noun)* expression describing the service.

**Inputs:** The user of a service will provide zero or more inputs. We also need a specification of what inputs are expected and which of these inputs are mandatory: "English" name for each input, and a semantic type for that input. For example: *Amount:* `Integer` specifies that we have an input called *Amount* of type `Integer` and *Part:* `PartName` specifies that we have an input called *Part* of type `PartName` (which could be an enumerated type).

**Outputs:** Each service must specify the outputs that it provides and each output is specified in the same way as an input.

**Attributes:** In addition, services may have attributes associated with them. Examples of such attributes include cost (for using the service), average response time for requests to that service, etc.

**Definition 5.6 (Type/Type Hierarchy $(\mathcal{T}, \leq)$)**
*A* type $\tau$ *is a set whose elements are called "values" of* $\tau$*. The pair* $(\mathcal{T}, \leq)$ *is called a* type hierarchy *if* $\mathcal{T}$ *is a set of types and* $\leq$ *is a partial ordering on* $\mathcal{T}$*.*

Figure 5.11 provides a hierarchy associated with the three motivating examples.

# Type Hierarchy

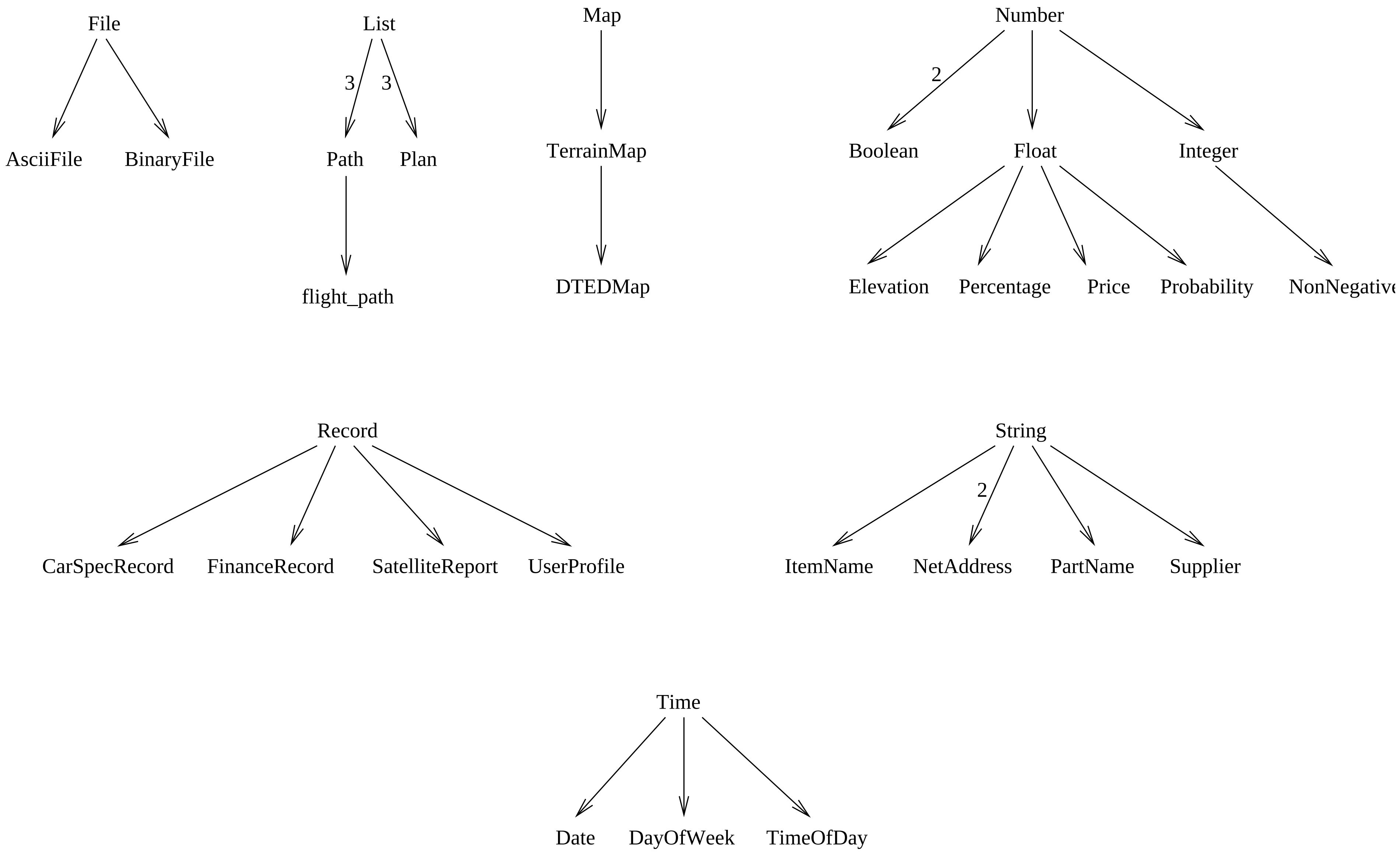



Figure 5.11: Example Type Hierarchy

**Definition 5.7 (Set of Type Variables $V_{\mathcal{T}}$)**
*Associated with any type hierarchy* $(\mathcal{T}, \leq)$, *is a set* $V_{\mathcal{T}}$ *of symbols called* type variables.

Intuitively, a type variable ranges over the values of a given type. For instance, `PartName` may be a type variable ranging over strings. When specifying the inputs required to invoke a service, we need to specify variables and their associated types. This is done in the usual way, as defined below.

**Definition 5.8 (Items $s : \tau$)**
*If* $s$ *is a variable ranging over objects of type* $\tau$, *then* $s : \tau$ *is called an* item.

> $s : \tau$ may be read as saying
> *"the variable* $s$ *may assume values drawn from the type* $\tau$*"*.

**Definition 5.9 (Item Atom)**
*If* $s\!:\tau$ *is an item, then* $\langle\texttt{I}\rangle s\!:\tau\langle\backslash\texttt{I}\rangle$ *(resp.* $\langle\texttt{MI}\rangle s\!:\tau\langle\backslash\texttt{MI}\rangle$ *) is called an* input *(resp.* mandatory input*) item atom, and* $\langle\texttt{O}\rangle s\!:\tau\langle\backslash\texttt{O}\rangle$ *is called an* output *item atom.*

Each input item is either ***mandatory*** or not. For example, $\langle\texttt{MI}\rangle$*Location:* `String`$\langle\backslash\texttt{MI}\rangle$ is a mandatory input item atom, while $\langle\texttt{I}\rangle$*Nogo:* `TerrainMap`$\langle\backslash\texttt{I}\rangle$ is a non-mandatory input item atom. The following are all valid output item atoms: $\langle\texttt{O}\rangle$*Path1:* `Path`$\langle\backslash\texttt{O}\rangle$, $\langle\texttt{O}\rangle$*Specs:* `CarSpecRecord`$\langle\backslash\texttt{O}\rangle$ and $\langle\texttt{O}\rangle$*Financing_plan:* `FinanceRecord`$\langle\backslash\texttt{O}\rangle$.

**Definition 5.10 (Service Description)**
*Let $sn$ be a service name, $i_1, \ldots, i_n$ be input item atoms, $mi_1, \ldots, mi_k$ be mandatory input item atoms, and $o_1, \ldots, o_r$ be output item atoms. Then,*

$$\begin{array}{ll} \langle \mathtt{S} \rangle & sn \\ & mi_1 \ldots mi_k \\ & i_1 \ldots i_n \\ & o_1 \ldots o_r \\ \langle \backslash \mathtt{S} \rangle & \end{array}$$

*is called a* service description.

**Definition 5.11 (Item List)**

*If* $s_1 : \tau_1, \ldots, s_n : \tau_n$ *are* $n \geq 1$ *items, then* $\langle \mathtt{I} \rangle s_1 : \tau_1, \ldots, s_n : \tau_n \langle \backslash \mathtt{I} \rangle$ *is an* input item list, *which is a shorthand for* $\langle \mathtt{I} \rangle s_1 : \tau_1 \langle \backslash \mathtt{I} \rangle \cdots \langle \mathtt{I} \rangle s_1 : \tau_n \langle \backslash \mathtt{I} \rangle$*; also,* $\langle \mathtt{MI} \rangle s_1 : \tau_1, \ldots, s_n : \tau_n \langle \backslash \mathtt{MI} \rangle$ *and* $\langle \mathtt{O} \rangle s_1 : \tau_1, \ldots, s_n : \tau_n \langle \backslash \mathtt{O} \rangle$ *are* mandatory input item lists *and* output item lists, *respectively, which are shorthands for the items* $\langle \mathtt{MI} \rangle s_1 : \tau_1 \langle \backslash \mathtt{MI} \rangle \cdots \langle \mathtt{MI} \rangle s_1 : \tau_n \langle \backslash \mathtt{MI} \rangle$ *and* $\langle \mathtt{O} \rangle s_1 : \tau_1 \langle \backslash \mathtt{O} \rangle \cdots \langle \mathtt{O} \rangle s_1 : \tau_n \langle \backslash \mathtt{O} \rangle$*, respectively.*

$\langle$`S`$\rangle$ *classify: user*

$\langle$`MI`$\rangle$*ssn:* `String`$\langle$`\MI`$\rangle$

$\langle$`I`$\rangle$*name:* `String`$\langle$`\I`$\rangle$

$\langle$`O`$\rangle$*class:* `UserProfile`$\langle$`\O`$\rangle$

$\langle$`\S`$\rangle$

This service may take as input, the name and the social security number of a user, and provide as output, a classification of the user as a "low," "medium," "high," or "very high" spender. The social security number is a mandatory input, whereas the **name** is **optional** as it can be uniquely determined from a person's social security number.

$\langle\texttt{S}\rangle$ *create: plan(flight)*

$\langle\texttt{MI}\rangle$*Location:* $\texttt{SatelliteReport}$, *Flightroute:* $\texttt{Path}$, *Nogo:* $\texttt{Map}\langle\backslash\texttt{MI}\rangle$

$\langle\texttt{O}\rangle$*Plan:* $\texttt{Plan}\langle\backslash\texttt{O}\rangle$

$\langle\backslash\texttt{S}\rangle$

This service takes three mandatory inputs (the location of the plane, the allocated flight route of the plane, and a set of Nogo areas), and generates a modified flight path for the plane.

$\langle$S$\rangle$ *monitor: availablestock*

$\langle$MI$\rangle$*Amount:* Integer, *Partid:* String$\langle$\MI$\rangle$

$\langle$I$\rangle$*Name:* String$\langle$\I$\rangle$

$\langle$O$\rangle$*Status:* String$\langle$\O$\rangle$

$\langle$\S$\rangle$

This service takes the *Amount* and *Part_id* of the requested part as mandatory inputs, and the *name* of the requested part as an optional input. The *Name* of a *part* maybe determined from its *Part_id*. This service returns as output the string `amount_available` or `amount_not_available`.

### 5.4.2 Metric and Matchmaking

Up to now, we defined distances between verbs and between noun-terms. But we need to have a **distance between service-names**!

**Definition 5.12 (Composite Distance Function cd)**
*Suppose we have two different sets of words* $\Sigma_1$ *and* $\Sigma_2$ *with* $\Sigma_1$*-hierarchy* $\mathcal{SH}_1 =_{def} (T_1, E_1, \wp_1)$ *and* $\Sigma_2$*-hierarchy* $\mathcal{SH}_2 =_{def} (T_2, E_2, \wp_2)$*. Let* $\mathbf{d}_1, \mathbf{d}_2$ *be the distance functions induced by* $\mathcal{SH}_1, \mathcal{SH}_2$*, respectively. Consider two pairs of words,* $\langle w_1, w_1' \rangle, \langle w_2, w_2' \rangle \in \Sigma_1 \times \Sigma_2$*. A* composite distance *function* **cd** *is any mapping from* $(\Sigma_1 \times \Sigma_2) \times (\Sigma_1 \times \Sigma_2)$ *to* $\mathbb{Z}^+$ *such that:*

1. $\mathbf{cd}(\langle w_1, w_1' \rangle, \langle w_2, w_2' \rangle) = \mathbf{cd}(\langle w_2, w_2' \rangle, \langle w_1, w_1' \rangle)$ *(Symmetry)*
2. $\mathbf{cd}(\langle w_1, w_1' \rangle, \langle w_1, w_1' \rangle) = 0$ *(Ipso-distance)*
3. *If* $\mathbf{d}_1(w_1, w_2) \leq \mathbf{d}_1(w_1, w_3)$, *then*
   $\mathbf{cd}(\langle w_1, w_1' \rangle, \langle w_2, w_2' \rangle) \leq \mathbf{cd}(\langle w_1, w_1' \rangle, \langle w_3, w_2' \rangle)$ *(Expansion of* $\mathbf{d}_1$*)*
4. *If* $\mathbf{d}_2(w_1', w_2') \leq \mathbf{d}_2(w_1', w_3')$, *then*
   $\mathbf{cd}(\langle w_1, w_1' \rangle, \langle w_2, w_2' \rangle) \leq \mathbf{cd}(\langle w_1, w_1' \rangle, \langle w_2, w_3' \rangle)$ *(Expansion of* $\mathbf{d}_2$*)*
5. $\mathbf{cd}(\langle w_1, w_1' \rangle, \langle w_3, w_3' \rangle) \leq \mathbf{cd}(\langle w_1, w_1' \rangle, \langle w_2, w_2' \rangle) +$
   $\mathbf{cd}(\langle w_2, w_2' \rangle, \langle w_3, w_3' \rangle)$ *(Triangle Inequality).*

**Example 5.1 (Composite Distances)**
*Let* $\mathbf{d}_1$ *and* $\mathbf{d}_2$ *be distances defined as in Section 5.2 on the verb and noun-term hierarchies given in Figure 141 on page 141 and Figure 142 on page 142, respectively. Moreover, let the composite distance function be defined as*

$$\mathbf{cd}(\langle w_1, w'_1\rangle, \langle w_2, w'_2\rangle) =_{def} \mathbf{d}_1(w_1, w_2) + \mathbf{d}_2(w'_1, w'_2).$$

*Now consider the following two pairs:* $\langle provide, information\rangle$ *and* $\langle broadcast, data(GPS)\rangle$*. As can be seen from Figure 141 on page 141, the distance between provide and broadcast is*

$$\mathbf{d}_1(provide, broadcast) = 2,$$

*as is the distance between information and data(GPS) (see Figure 142 on page 142).*

Thus, the composite distance between these two pairs is given by

$$\begin{aligned}
&\mathbf{cd}(\langle \textit{provide}, \textit{information} \rangle, \\
&\quad \langle \textit{broadcast}, \textit{data(GPS)} \rangle) &=& \ \mathbf{d}_1(\textit{provide}, \textit{broadcast}) + \mathbf{d}_2(\textit{information}, \textit{data(GPS)}) \\
& &=& \ 4.
\end{aligned}$$

As another example, consider the pairs $\langle \textit{identify}, \textit{items} \rangle$ and $\langle \textit{determine}, \textit{product} \rangle$. In this case, as given by Figure 141 on page 141, the distance $\mathbf{d}_1(\textit{identify}, \textit{determine})$ between *identify* and *determine* is 5. And from Figure 142 on page 142, the distance between *items* and *product* is $\mathbf{d}_2(\textit{items}, \textit{product}) = 1$. Then, the composite distance between $\langle \textit{identify}, \textit{items} \rangle$ and $\langle \textit{determine}, \textit{product} \rangle$ will be the sum of their verb and nounterm distances, i.e., 6.

What if we are looking for the distance between $n_1(n'_1)$ and $n_2(n'_2)$, but these terms do not occur (only $n_1, n'_1, n_2, n'_2$ are in nt)?

Then we use **cd** where $\Sigma_1 := \Sigma_2 :=$ nt.

What if we are looking for the distance between $n_1(n_2)$ and $n_1$ but the term $n_1(n_2)$ does not ocurr? (There might be a synonym $n_3$ for $n_2$ s.t. $n_1(n_3)$ ocurrs.)

Then we roughly estimate: see Definition of $\mathbf{d_G}$.

**Definition 5.13 (The Function $\mathbf{d_G}$)**

*We interpret, $n_1$ as $n_1(general)$, e.g., information as information(general), and assume that a function denoted by $\mathbf{d_G}$ for computing the distance between any noun $n$ and general is given to the system. E.g.: ($w_1, \ldots, w_k$ are the weights of all edges between the Noun-Term-node and any of its neighboring vertices)*

$$\mathbf{d_G}(n, general) =_{def} max(w_1, \ldots, w_k).$$

**Example 5.2 (Distances)**
*When $n$ is the noun-term map or navigation, $\mathbf{d_G}(n, general) = 2$ but when $n$ is plan or route, $\mathbf{d_G}(n, general) = 1$.*

*Consider a query which asks for map(region). Which noun-term should we consider first?*

*Although there is no noun-term in our hierarchy named map(region), there are noun-terms for both map and region. Recall that $\mathbf{d_G}(map, general) = 2$. If we can find a noun-term $n$ with a distance of 2 or less from map(region), we should start at $n$. Otherwise, we should start at map.*

*In our current example, we should start at map(area) as region has a distance of 1 from area and so map(area) has a distance of $1 < 2$. However, if we were looking for map(city), there is no noun-term with a distance of 2 or less so we should start at map.*

### Matchmaking

It is easy to define **find_nn**: An algorithm to solve the $k$-nearest neighbor problem.

Given a pair $\langle v, nt \rangle$ specifying a desired service, this algorithm will return a set of $k$ agents that provide the most closely matching services.

Closeness between $\langle v, nt \rangle$ and another pair $\langle v', nt' \rangle$ is determined by using

1. the distance functions associated with the verb and noun-term hierarchies,
2. a composite distance function **cd** specified by the agent invoking the **find_nn** algorithm.

## Range Computations

The **range** Algorithm answers queries of the form

*"Find all agents that provide a service* $vnt = \langle V', NT' \rangle$ *which is within a distance D of a requested service* $vnt = \langle V, NT \rangle$*".*

### 5.4.3 Simulation Results

We are interested in

- the efficiency of finding *similar* services and
- the quality of the *matching* services provided as the output.

#### Performance Results

Based on a *NASA* hierarchy consisting of 17,445 words (for experimental purposes, the same hierarchy was used as both a verb and a noun hierarchy, although the *IMPACT* prototype uses different hierarchies). Weights on all edges in the hierarchies were assumed to be 1 and the composite distance function was taken to be `sum`.

The algorithms were implemented in C++ and the experiments were conducted on a Sun Sparc.

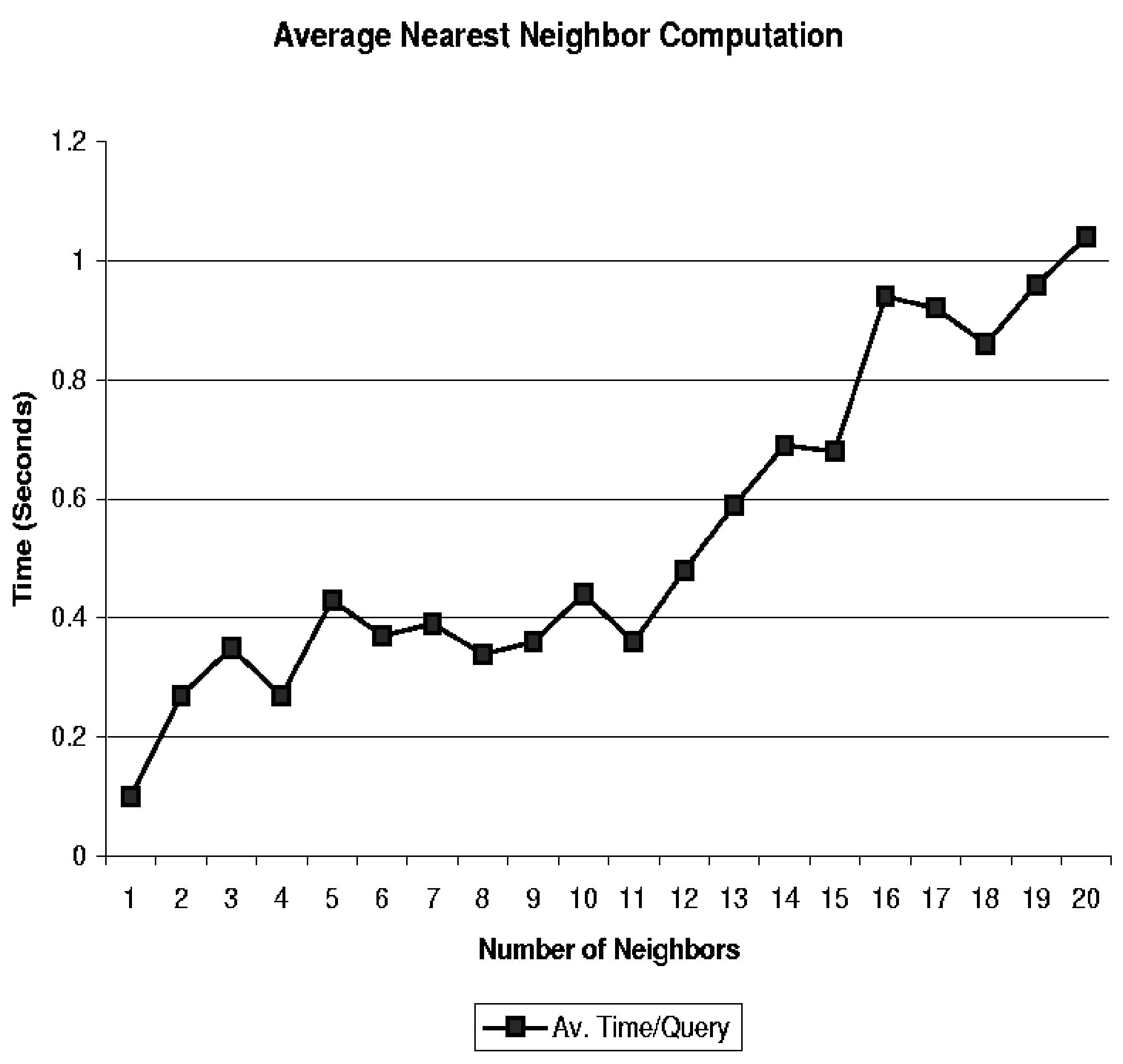



Figure 5.12: Performance of *k*-nearest neighbor algorithm, Average Time

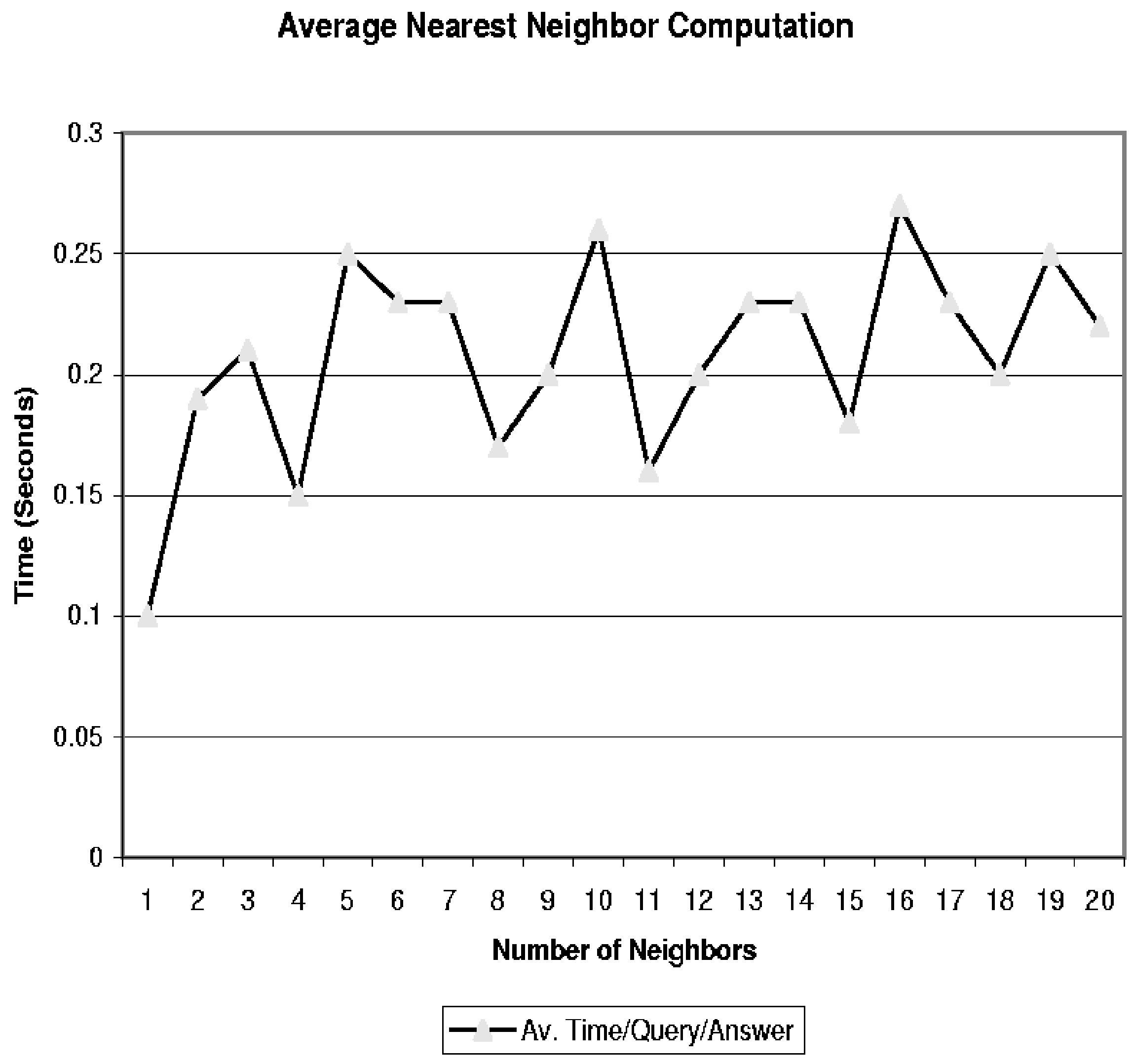



Figure 5.13: Performance of *k*-nearest neighbor, Average time per answer

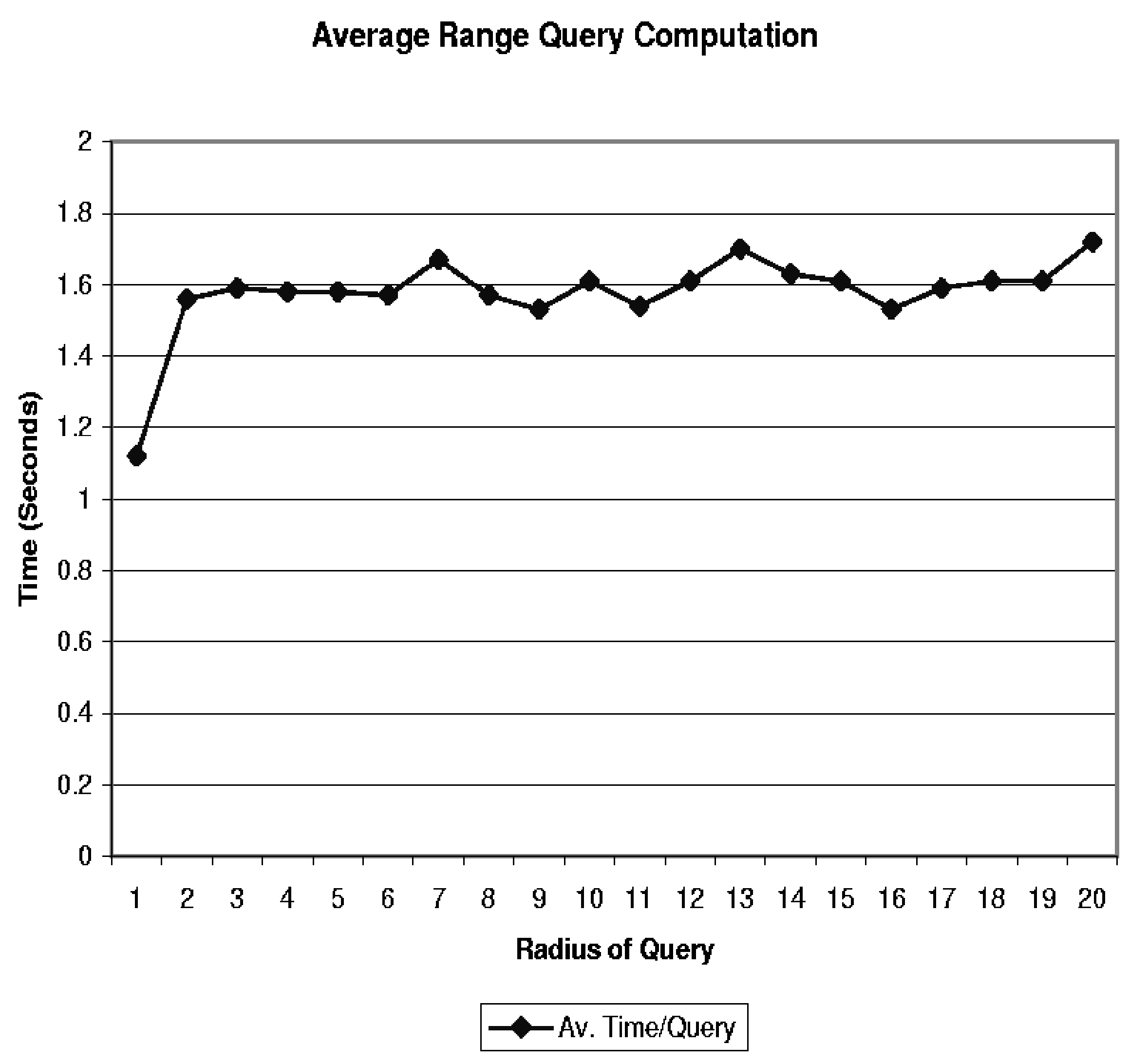



Figure 5.14: Performance of range query algorithm, Average Time

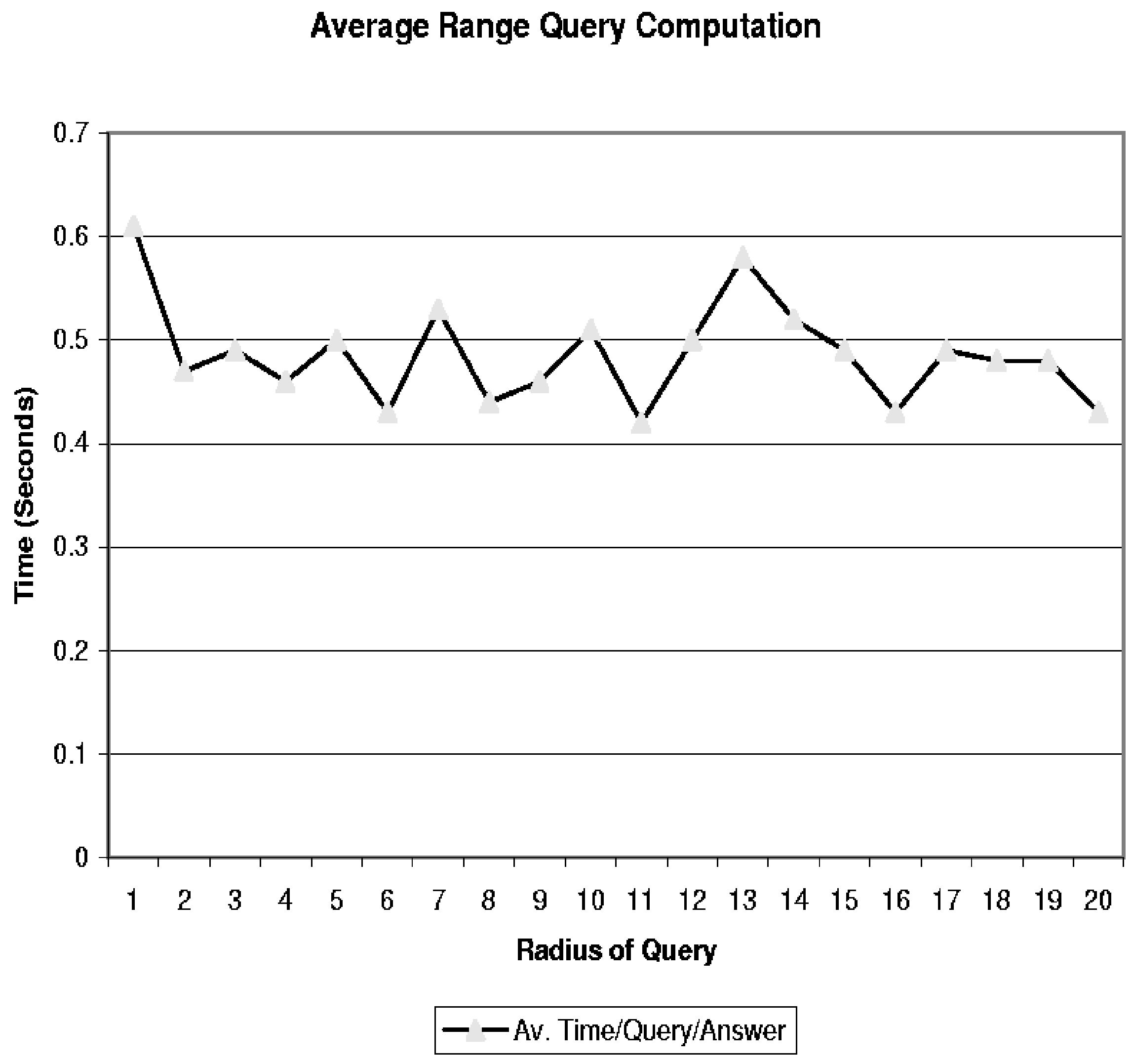
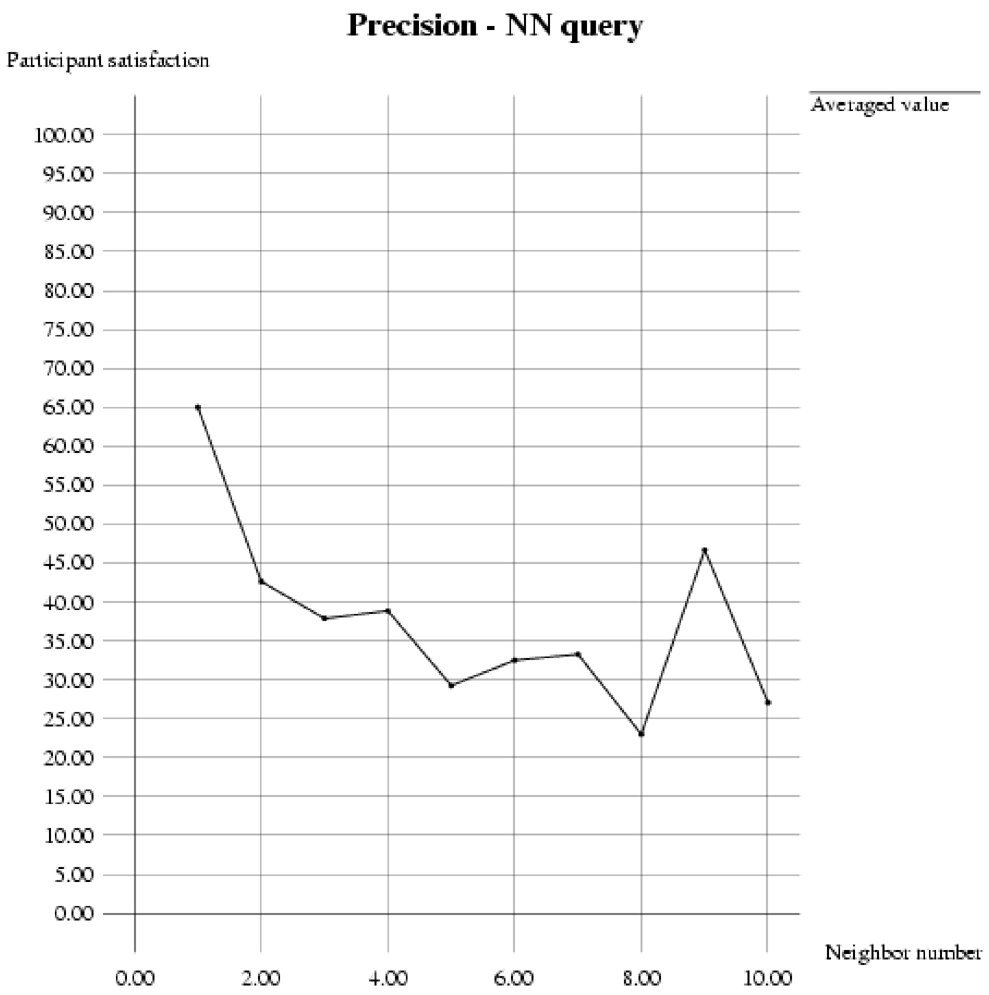



Figure 5.15: Performance of range query algorithm, Average time per answer

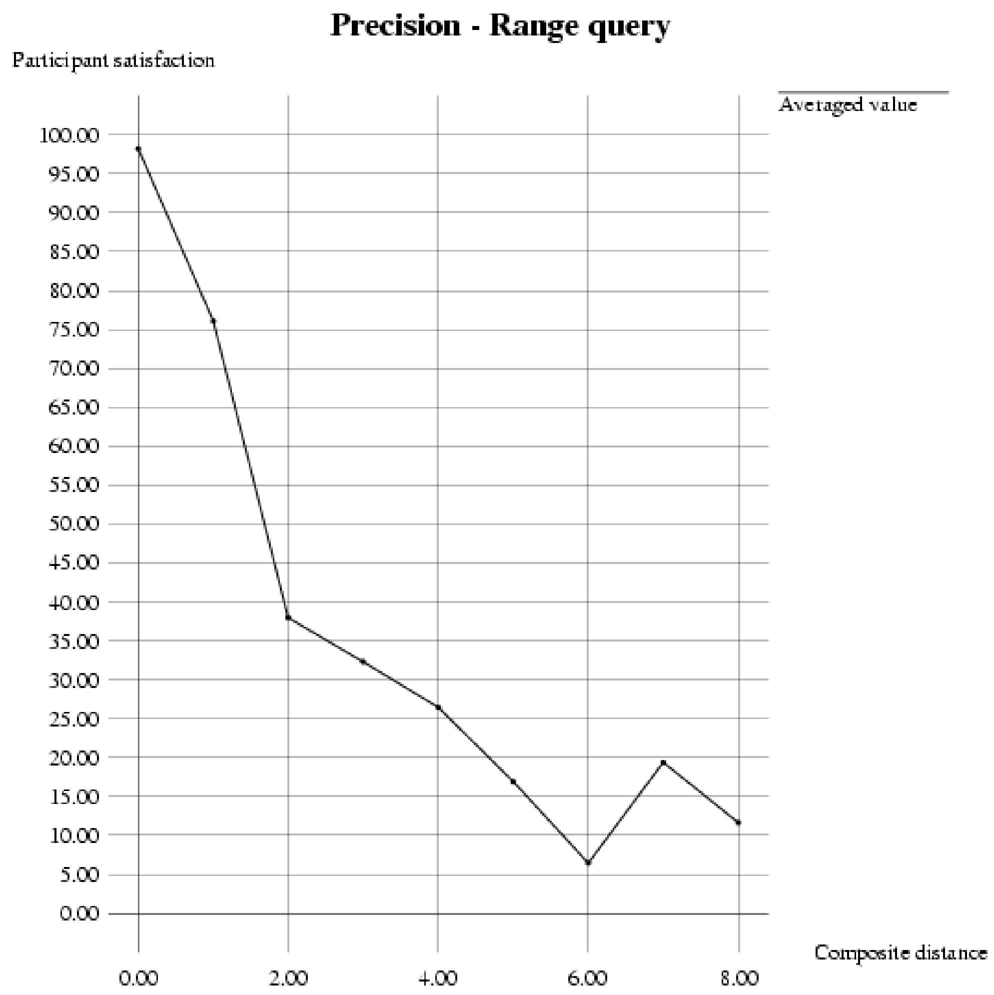
### Quality of Returned Matches

We conducted an experiment involving 35 participants:

1. We used a simple verb hierarchy (10 nodes), noun-term hierarchy (90 nodes), and ServiceTable (100 services).
2. After an initial training phase, participants entered a precision phase where they were asked to perform 10 nearest neighbor and 10 range queries of their choice.
3. After each query result, participants typed in a ranking between 0 (least satisfied) and 100 (most satisfied).
4. Average satisfaction for nearest neighbor and range queries are shown below.




Precision for the *k*-nearest neighbor algorithm

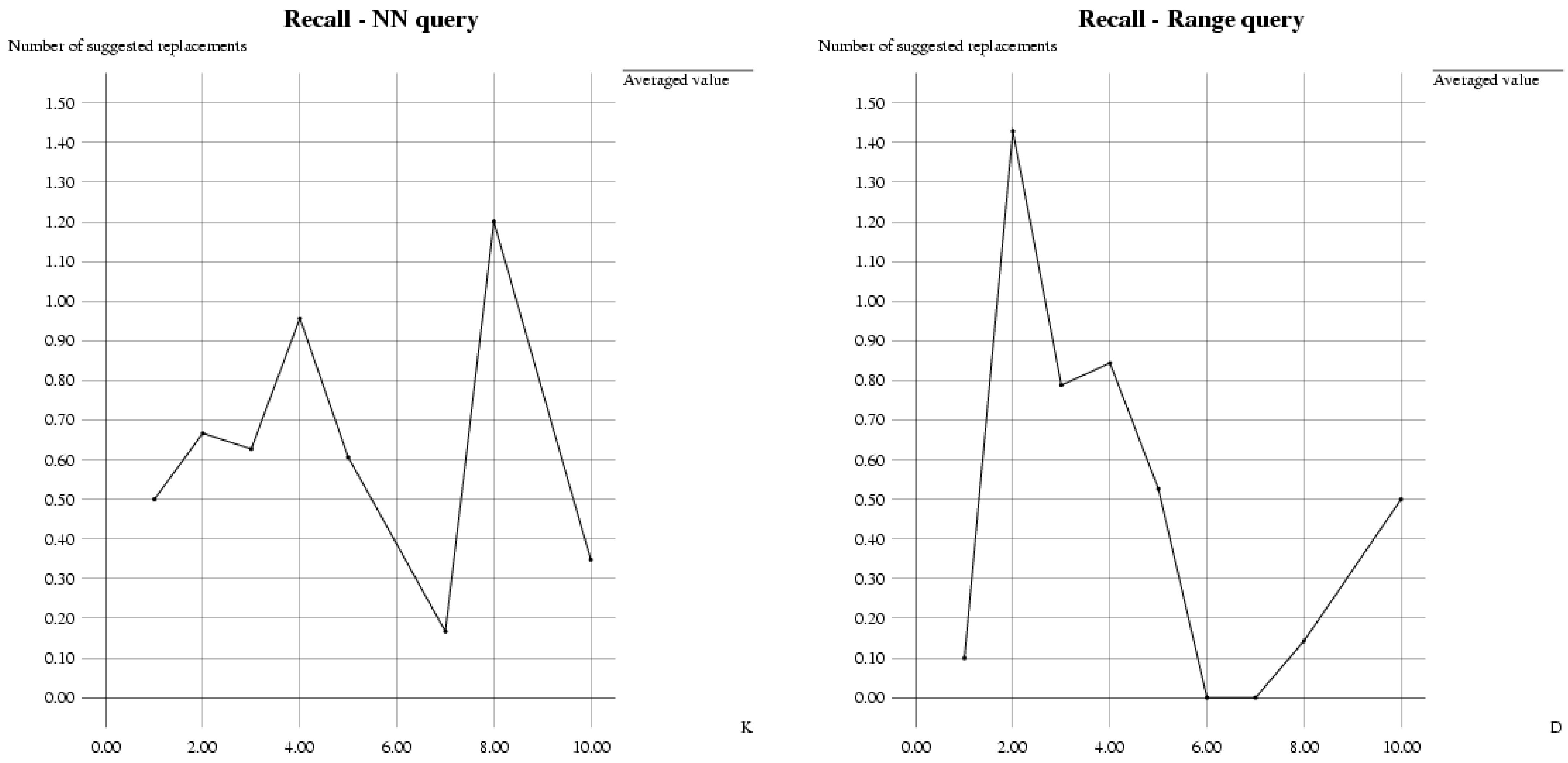



Precision for the range algorithm

Figure 5.16: Experimental Results of Precision of our Algorithms

After completing the precision phase, participants started the **recall phase**.

1. They were allowed to view the ServiceTable (which up to this point was not available to them).
2. Meanwhile, they were presented with text boxes containing the query answers they gave in the previous phase.
3. After each answer, they were instructed to type in the name of all services in ServiceTable which did not appear as a query result but which should have been returned as an answer.
4. The **average number of these "suggested replacements"** for nearest neighbor and range query answers are shown below.




Recall for the $k$-nearest neighbor algorithm Recall for the range algorithm (radius $D$)

Figure 5.17: Experimental Results of Recall of our Algorithms

## 5.5 Summary

1. We introduced **three szenarios** where multi-agency is important.
2. We presented the main *IMPACT*-architecture.
3. Agents need to use **services of other agents**.
   (a) We do not assume that agents precisely know about services of other agents.
   (b) We defined a language where such requests can be formulated ($\leadsto$ **service description language SDL**).
   (c) We presented algorithms to find the best matches for a request ($\leadsto$ **find_nn**, **range**) .

## References

Apt, K., H. Blair, and A. Walker (1988). Towards a Theory of Declarative Knowledge. In J. Minker (Ed.), *Foundations of Deductive Databases and Logic Programming*, pp. 89–148. Washington DC: Morgan Kaufmann.

Arens, Y., C. Y. Chee, C.-N. Hsu, and C. Knoblock (1993). Retrieving and Integrating Data From Multiple Information Sources. *International Journal of Intelligent Cooperative Information Systems 2*(2), 127–158.

Arisha, K., F. Ozcan, R. Ross, V. S. Subrahmanian, T. Eiter, and S. Kraus (1999, March/April). IMPACT: A Platform for Collaborating Agents. *IEEE Intelligent Systems 14*, 64–72.

Bayardo, R., et al. (1997). Infosleuth: Agent-based Semantic Integration of Information in Open and Dynamic Environments. In J. Peckham (Ed.), *Proceedings of ACM SIGMOD Conference on Management of Data*, Tucson, Arizona, pp. 195–206.

Brink, A., S. Marcus, and V. Subrahmanian (1995). Heterogeneous Multimedia Reasoning. *IEEE Computer 28*(9), 33–39.

Chawathe, S., et al. (1994, October). The TSIMMIS Project: Integration of Heterogeneous Information Sources. In *Proceedings of the 10th Meeting of the Information Processing Society of Japan*, Tokyo, Japan. Also available via anonymous FTP from host db.stanford.edu, file /pub/chawathe/1994/tsimmis-overview.ps.

Dix, J., S. Kraus, and V. Subrahmanian (1999, September). Temporal agent programs. Technical Report CS-TR-4055, Dept. of CS, University of Maryland, College Park, MD 20752. currently under submission for a Journal.

Dix, J., M. Nanni, and V. S. Subrahmanian (2000). Probabilistic agent reasoning. *Transactions of Computational Logic 1*(2).

Dix, J., V. S. Subrahmanian, and G. Pick (2000). Meta Agent Programs. *Journal of Logic Programming 45*(1).

Eiter, T., V. Subrahmanian, and G. Pick (1999). Heterogeneous Active Agents, I: Semantics. *Artificial Intelligence 108*(1-2), 179–255.

Eiter, T., V. Subrahmanian, and T. J. Rogers (2000). Heterogeneous Active Agents, III: Polynomially Implementable Agents. *Artificial Intelligence 117*(1), 107–167.

Eiter, T. and V. S. Subrahmanian (1999). Heterogeneous Active Agents, II: Algorithms and Complexity. *Artificial Intelligence 108*(1-2), 257–307.

Genesereth, M. R. and S. P. Ketchpel (1994). Software Agents. *Communications of the ACM 37*(7), 49–53.

Rogers Jr., H. (1967). *Theory of Recursive Functions and Effective Computability*. New York: McGraw-Hill.

Subrahmanian, V., P. Bonatti, J. Dix, T. Eiter, S. Kraus, F. Özcan, and R. Ross (2000). *Heterogenous Active Agents*. MIT-Press.

Wiederhold, G. (1993). Intelligent Integration of Information. In *Proceedings of ACM SIGMOD Conference on Management of Data*, Washington, DC, pp. 434–437.

Wilder, F. (1993). *A Guide to the TCP/IP Protocol Suite*. Artech House.