Conceptual Graph Representation of a Work Order Allocation System*

Jeff Riley
Computer Science Department
Royal Melbourne Institute of Technology
GPO Box 2476V
Victoria, Australia
jriley@yallara.cs.rmit.edu.au

Abstract

A computer vendor operates a software help desk organisation. The help desk organisation needs a system to automatically allocate incoming work orders. Allocation of the work orders to engineers is to be done on the basis of engineer availability and skills. This paper will attempt to represent a simplified model of such a system using conceptual graphs.

1 Introduction

Requests for assistance from customers are recorded in a database. These requests are referred to as calls, and are initially unassigned (that is, the call has not been allocated to an engineer).

For the purpose of developing the simplified model, the following information is considered to be recorded for each call:

a unique identifier
an owner engineer if allocated
the product the customer believes the call pertains to
a free form text description of the problem

Some of this information will be recorded in compound structures.

Engineers in this organisation are grouped into teams based on engineer skills. Teams are the resource which logically handle calls: in practice any call handled by the team is immediately forwarded to an available engineer from the team. Engineers may belong to multiple teams, and teams may handle calls for whichever products its member engineers are skilled in

* This paper is part of the RMIT CS 433 Knowledge Representation Course assessment

Information known to the system is:

the set of skills associated with each engineer
the time at which each engineer was most recently assigned a call
a means of communicating with each available engineer
the set of engineers which belongs to any team
engineer location

Information recorded about each product includes a flag which determines whether the system should use the problem description to revise the product specified by the customer. For a call which pertains to a product whose flag isYES, the system will attempt to revise the product by analysing the problem description. If the system is able to determine a revised product from the problem description, the call is modified to reflect the revised product before any further processing. Typically, products with the flag set are very broad, generic categorisations, and the problem description is used to attempt to narrow the search for specific skills require to handle the call.

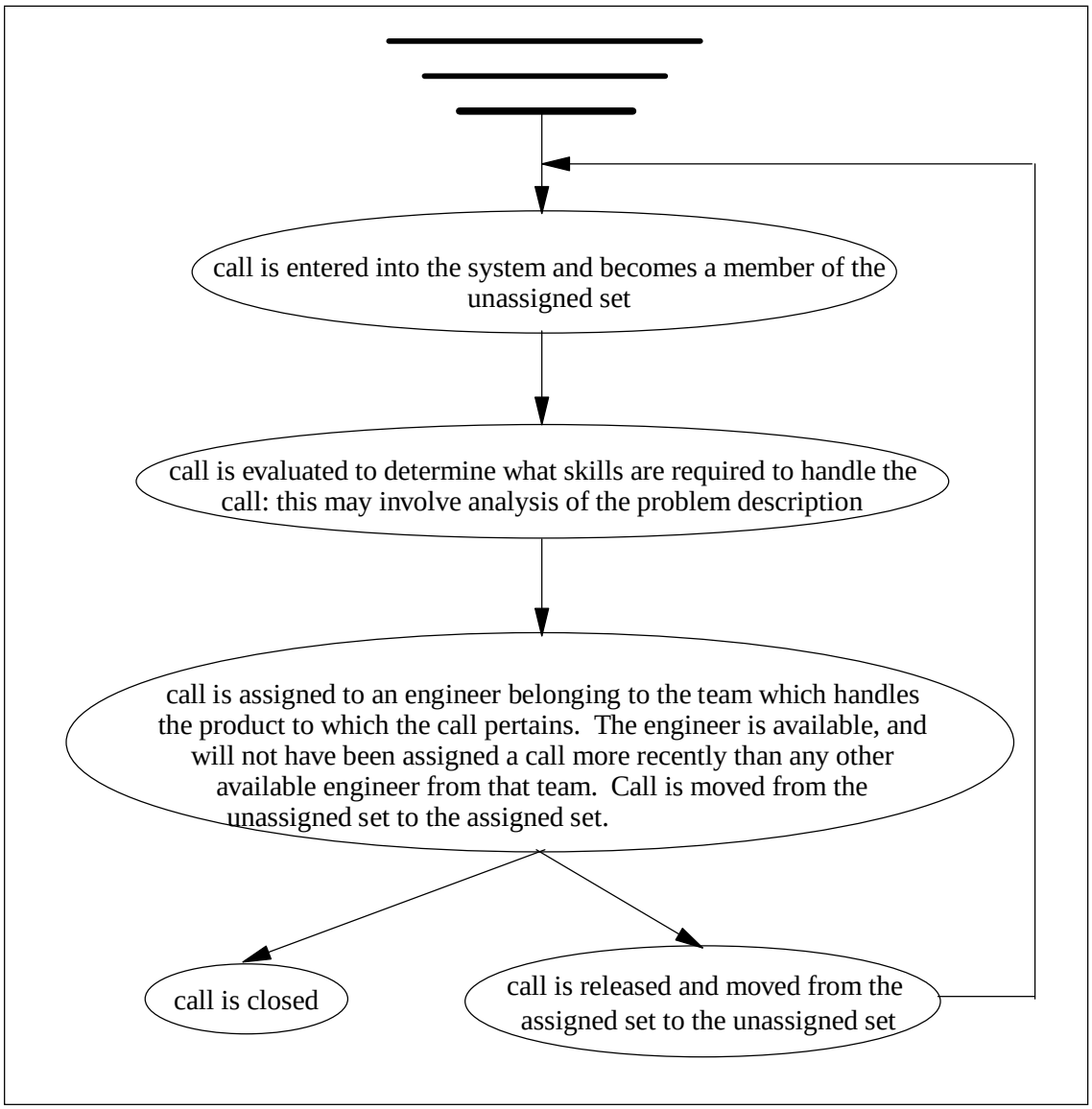
When a call is first recorded, and before it is assigned to an engineer, it belongs to the set of unassigned calls. Once assigned to an engineer, the call belongs to the set of assigned calls. Engineers are able to release a call from the assigned set back to the unassigned set. In this circumstance (in this simplified model), any such call is treated as a new call.

Once a call has been allocated (assigned) to an engineer, the engineer is notified of the allocation via electronic postnote on the engineer's workstation as well as by alphanumeric pager.

In modelling this simplified system the following events are considered:

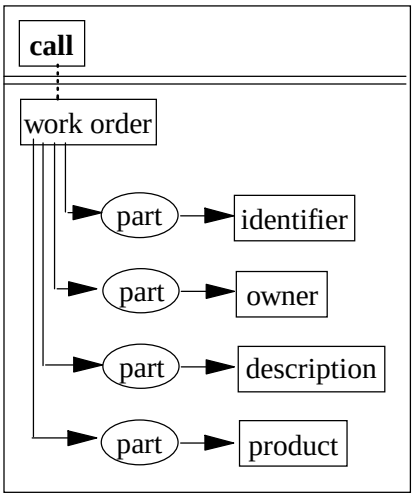
evaluation: analysis of the work order description to refine the product associated with the work order
assignment: determination and location of the most suitable engineer to handle the work order, and assignment of the work order to that engineer.
release: release of the work order by the owner engineer. This moves the work order back to the unassigned set.
notification: announcing to the owner engineer that a work order has been allocated to him/her.

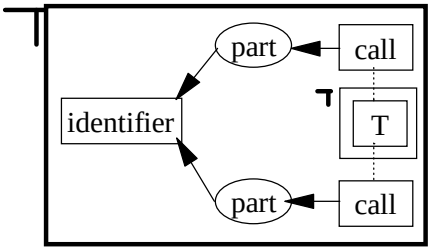
2 Data Flow


call is entered into the system and becomes a member of the unassigned set
call is evaluated to determine what skills are required to handle the call: this may involve analysis of the problem description
call is assigned to an engineer belonging to the team which handles the product to which the call pertains. The engineer is available, and will not have been assigned a call more recently than any other available engineer from that team. Call is moved from the unassigned set to the assigned set.
call is closed
call is released and moved from the assigned set to the unassigned set

3 Type Specifications

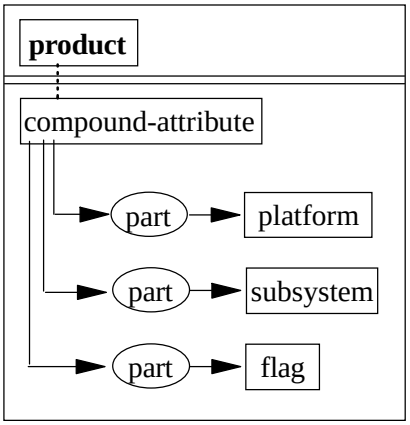
3.1 call





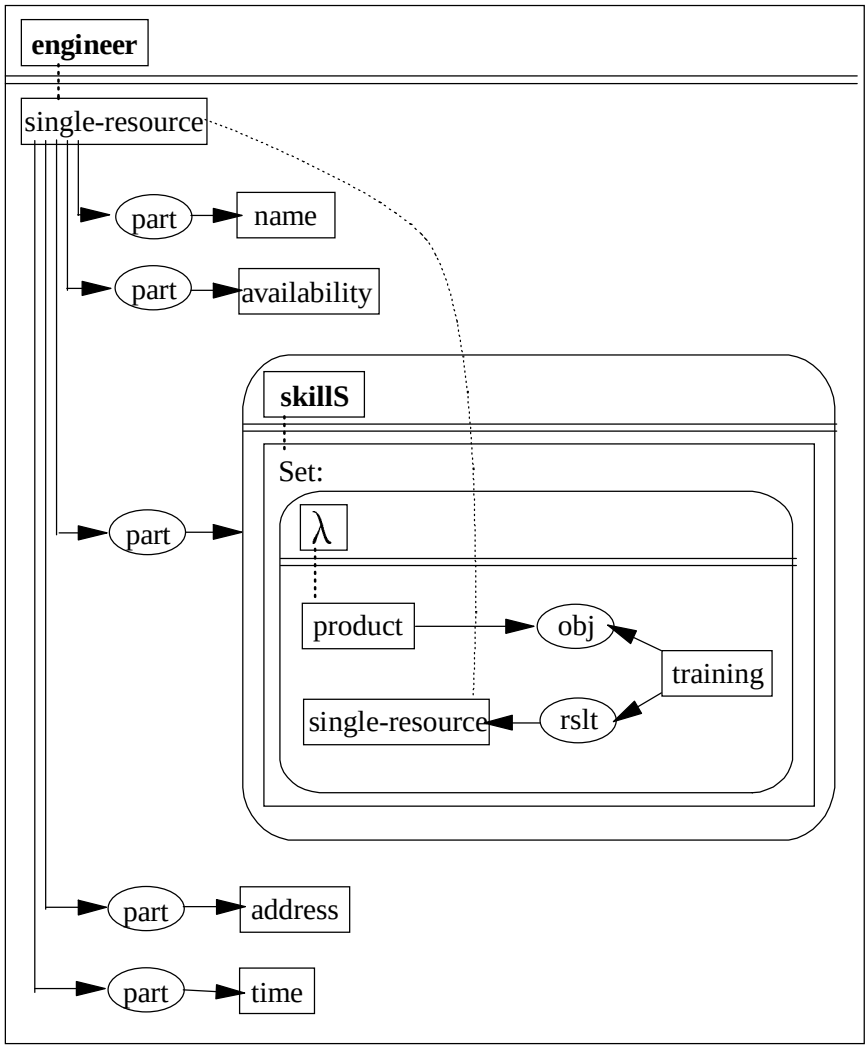
A **call** is a type of work order consisting of an **identifier**, an **owner**, a **description** and a **product**. The identifier uniquely identifies the work order. If present, the owner identifies the engineer to which the work order has been allocated. The description is a text description of the work to be carried out, and the product identifies which software product the work order pertains to. Call identifiers are unique.

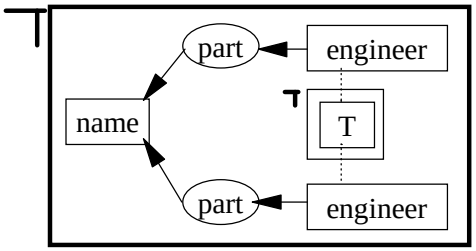
3.2 product



A **product** is a type of compound-attribute consisting of a **platform**, a **subsystem**, and a **flag**. The platform indicates the computer hardware involved in the work order, whereas the subsystem identifies the specific software involved. The flag indicates whether the work order description is required to further refine the subsystem.

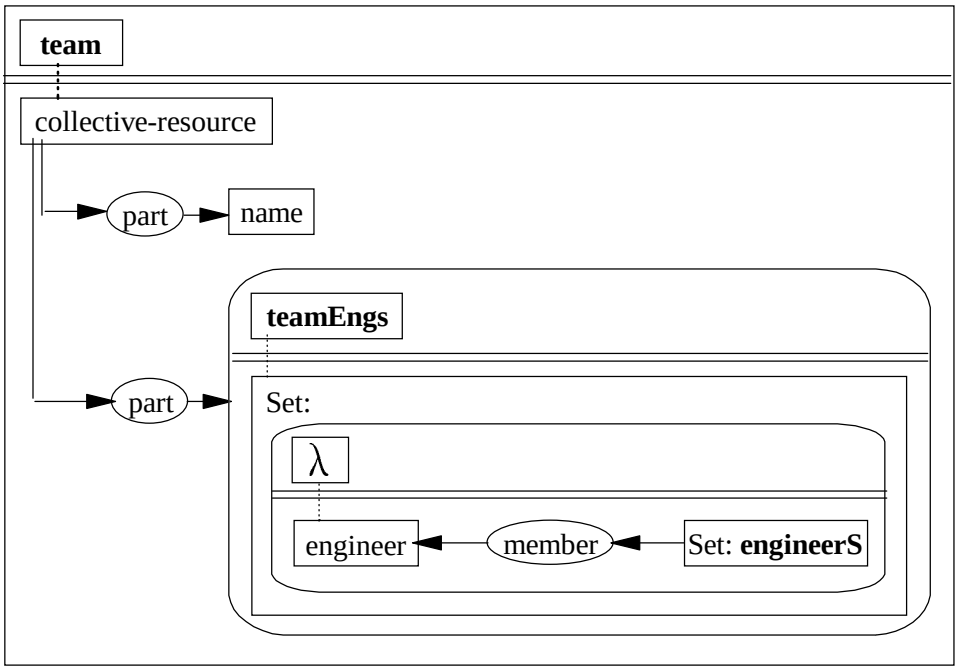
3.3 engineer





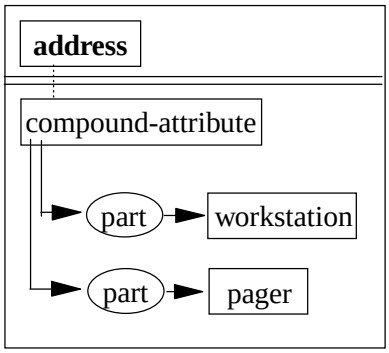
An **engineer** is a type of single-resource consisting of a **name**, an **availability,** a set of skills (**skillS)**, and **address** and a **time**. The availability is an indicator set by the engineer to show his/her availability to accept a new work order. The set of skills has as members those products for which the engineer has been trained. The address is the location of the engineer's workstation and pager number, and the time is the time at which the engineer was last assigned a call. Engineer names are unique.

3.4 team



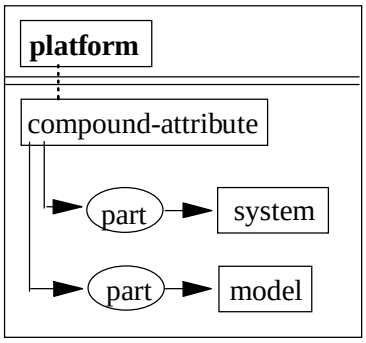
A **team** is a type of collective-resource consisting of a **name**, and a set of engineers (**engineerS**). The set of engineers has as members those engineer**s** which have been trained on any of the products to be handled by the team.

3.5 address



An **address** is a type of compound-attribute consisting of the network name of a **workstation,** and a **pager** number.

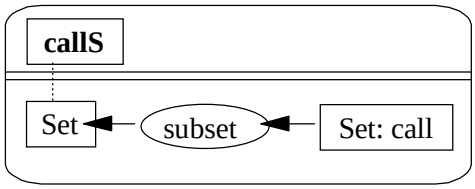
3.6 platform



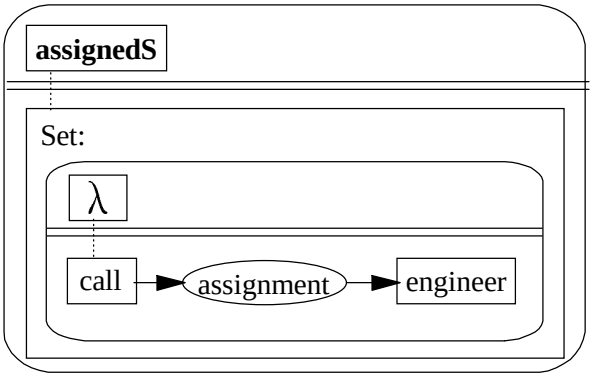
A **platform** is a type of compound-attribute consisting of the **system** and the **model.** The system is the generic hardware type associated with the work order, and the model is the specific hardware model.

4 Set Specifications

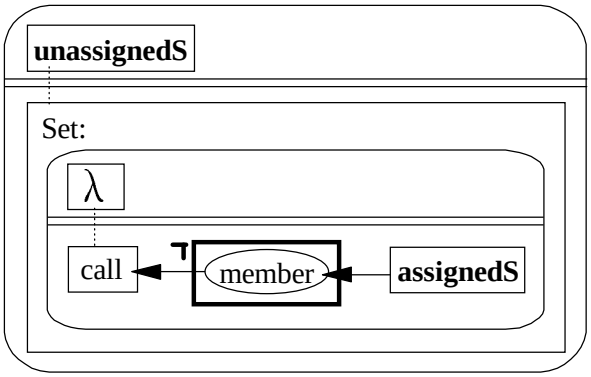
4.1 callS: The set of calls.



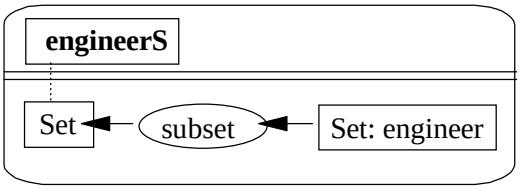
4.2 assignedS: The set of calls which have been assigned to an engineer.



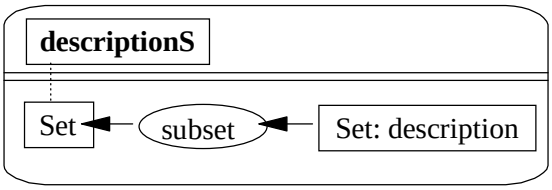
4.3 unassignedS: The set of calls which have not been assigned to an engineer.



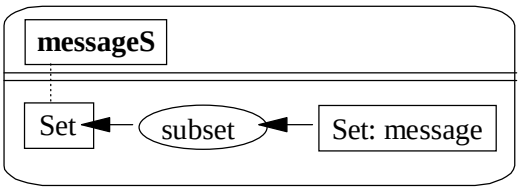
4.4 engineerS: The set of engineers.



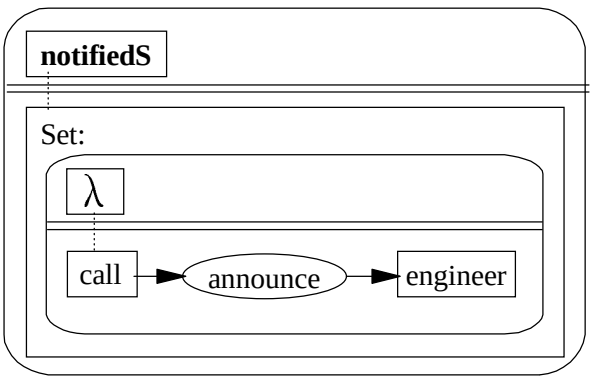
4.5 descriptionS: The set of call descriptions.



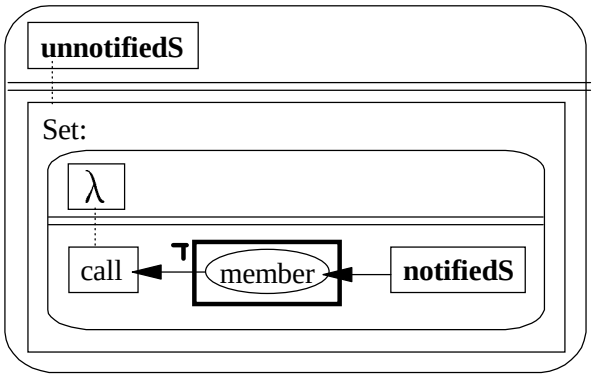
4.6 messageS: The set of notification messages.



4.7 notifiedS: The set of calls for which the owner engineers have been notified.

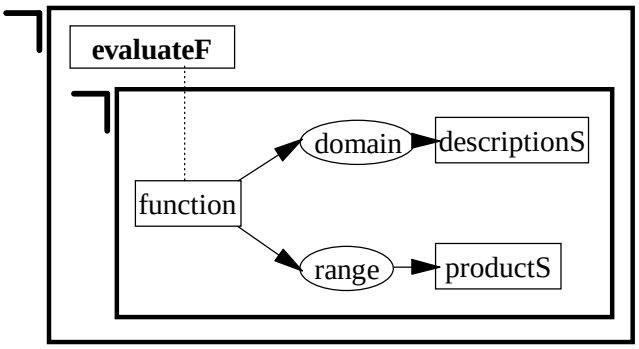


4.8 unnotifiedS: The set of assigned calls for which the owner engineers have not been notified.



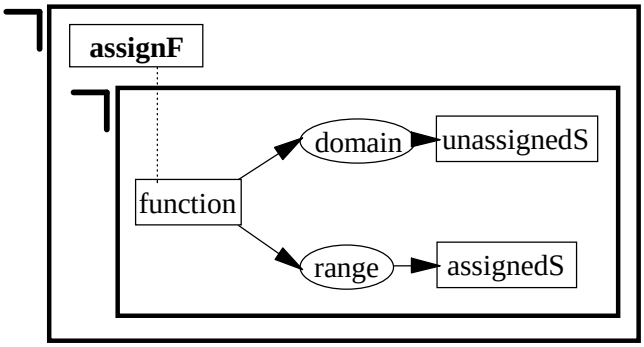
5 Function Declarations

5.1 evaluateF



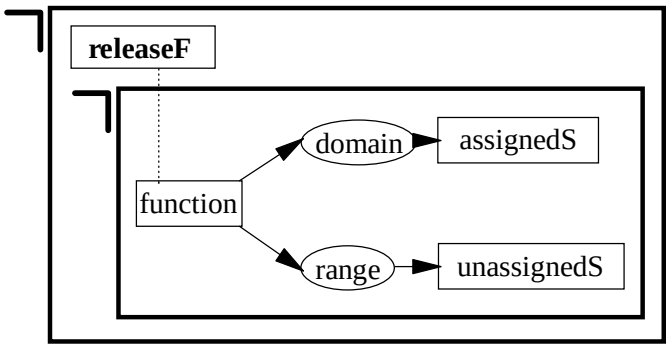
The evaluate function maps the set of descriptions to the set of products. Specifically, the function will determine from the description associated with a call, the product to which the call refers, and modify the call accordingly.

5.2 assignF



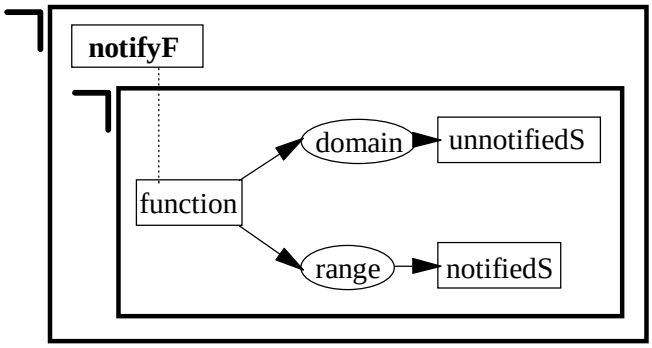
The assign function maps the set of unassigned calls to the set of assigned calls. Specifically, the function will locate an engineer with the skillset necessary for the call and modify the call such that the engineer located becomes the owner of the call. More detail is presented in the function specification.

5.3 releaseF



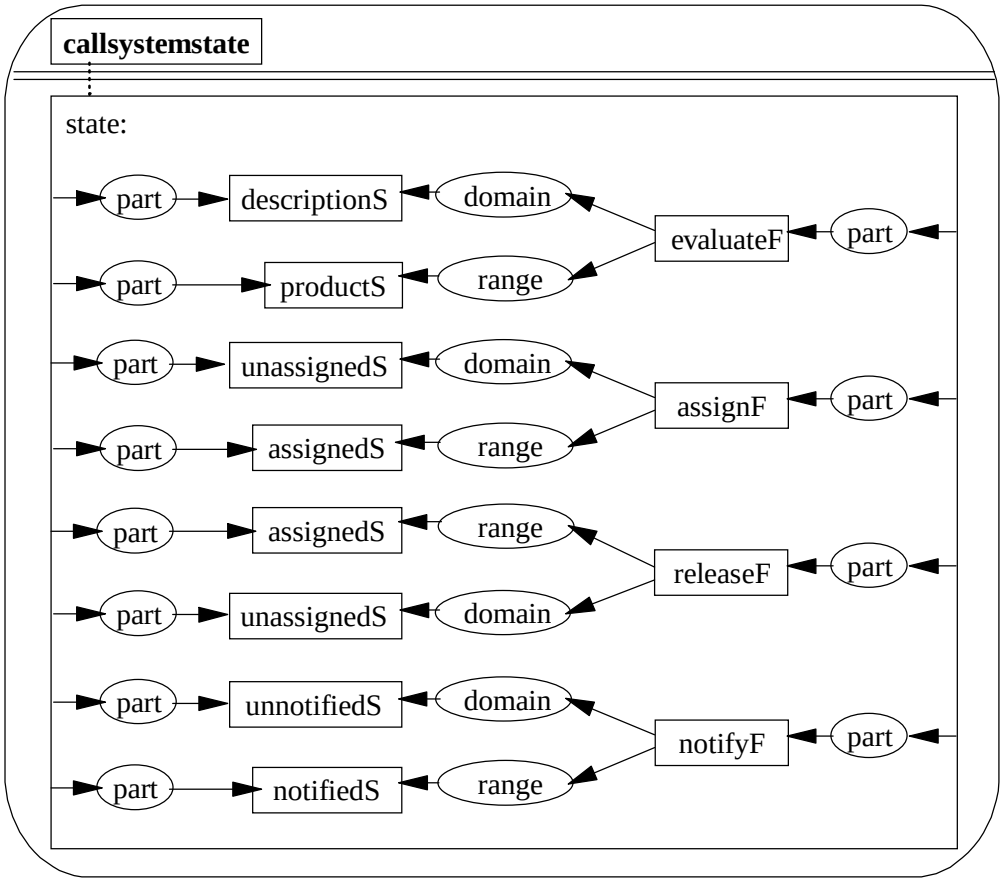
The release function maps the set of assigned calls to the set of unassigned calls. Specifically, the function will modify the call such that the call has no owner engineer.

5.4 notifyF

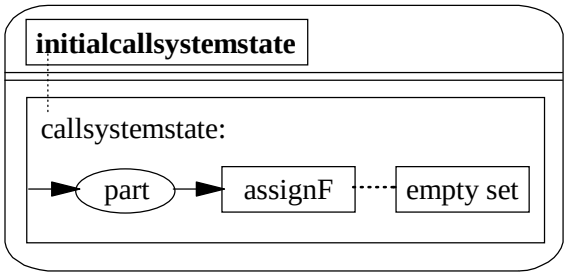


The assign function maps the set of unnotified calls to the set of notified calls. Specifically, the function will announce to the owner engineer that s/he has been allocated a new work order (call).

6 State Schema

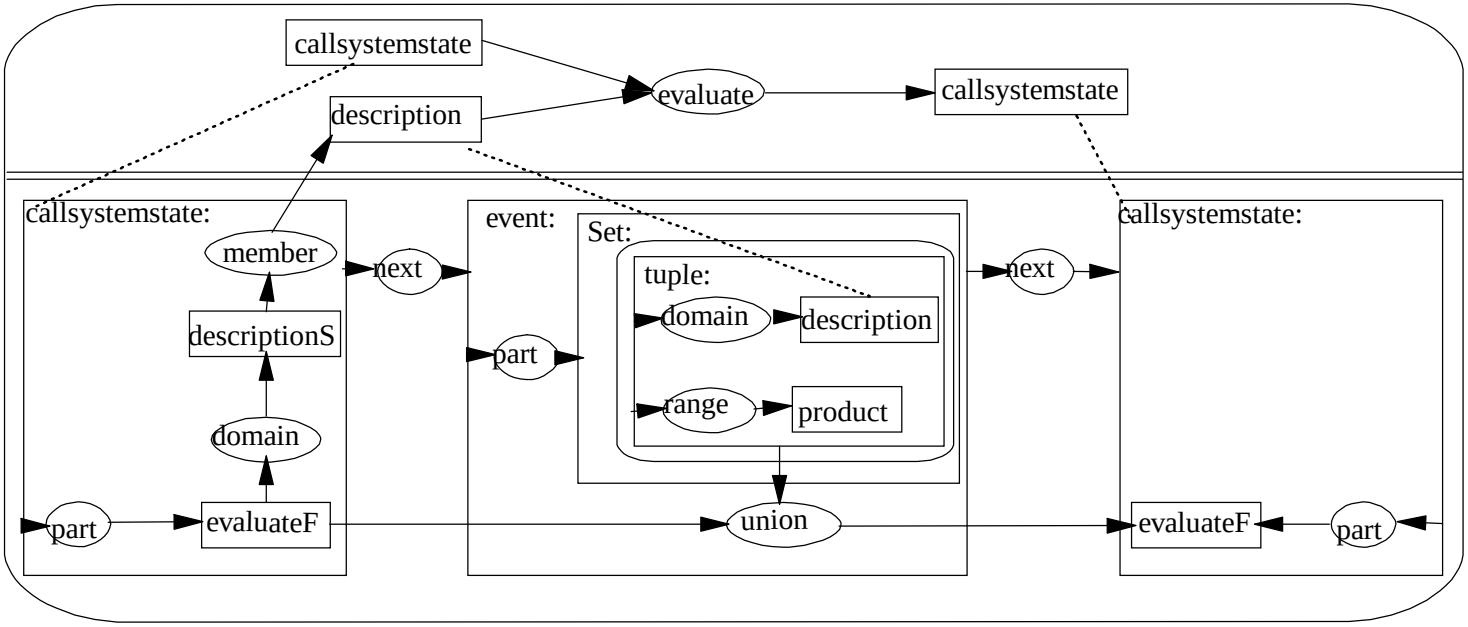


6.1 initial state



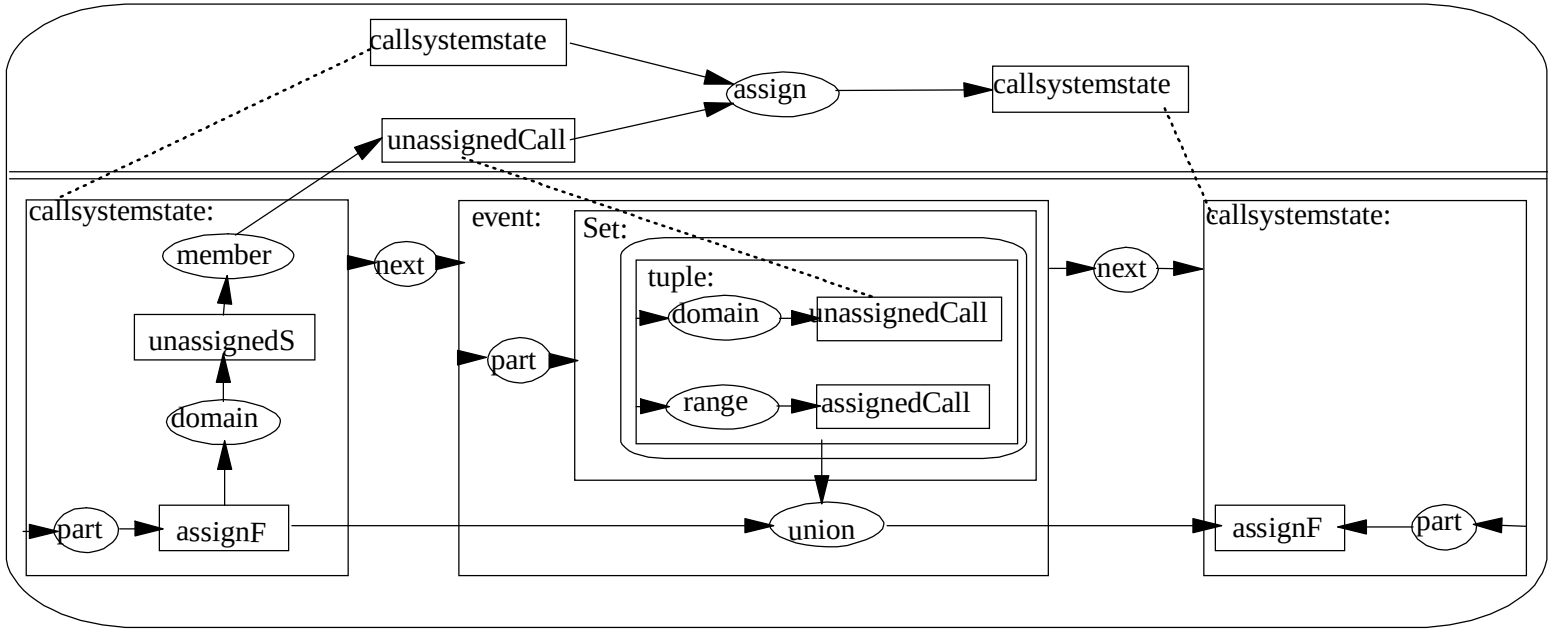
7 Operation Schema

7.1 call evaluation



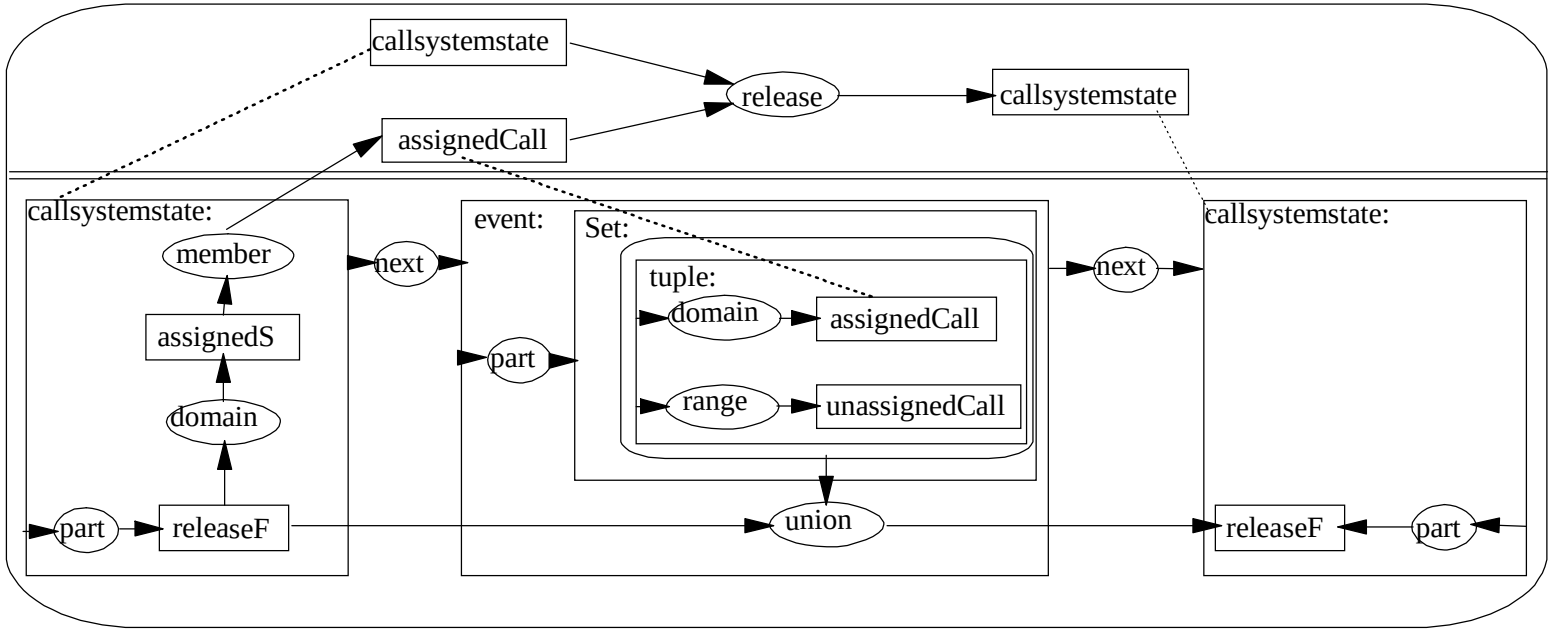
This schema specifies the evaluation of the description of a call.
The description is evaluated to determine a revised product for the call.
(see also the discussion of function evaluateF)

7.2 call assignment



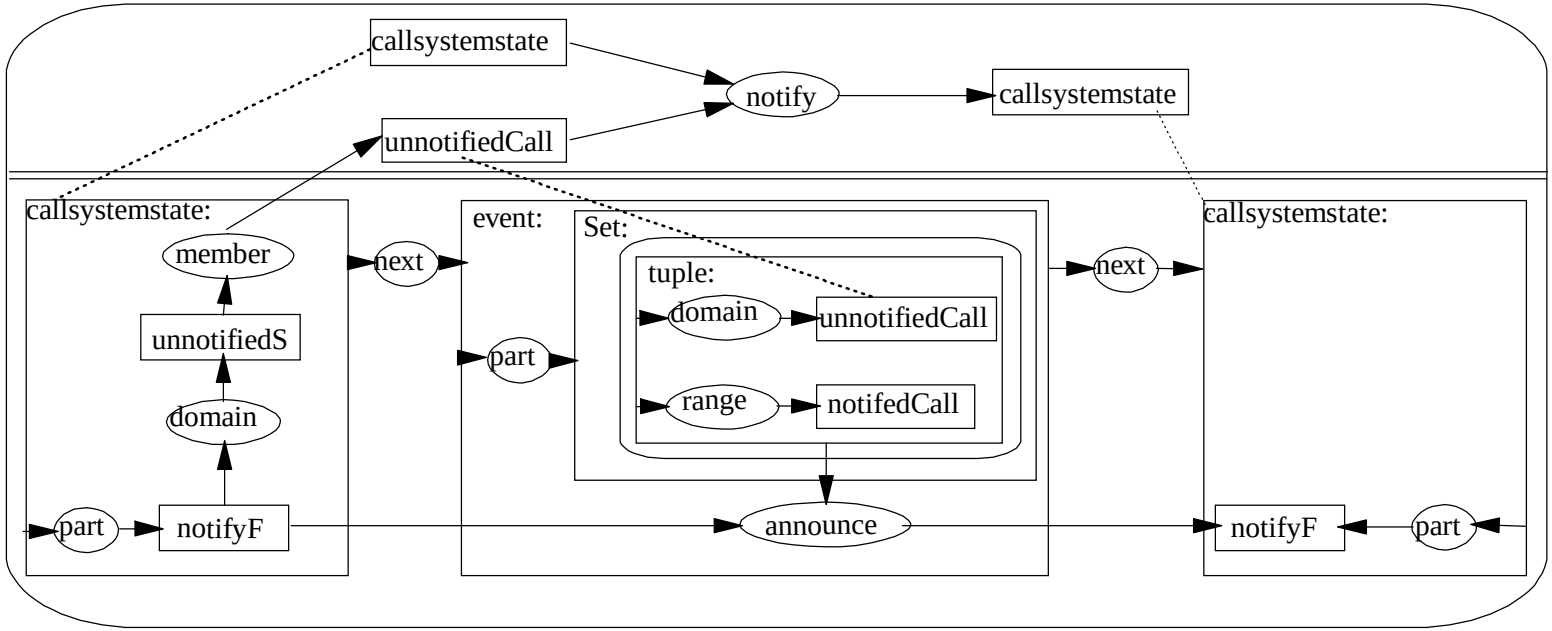
This schema specifies the assignment of a call.
An unassigned call is assigned to an engineer and is moved from the set of unassigned calls to the set of assigned calls.
(see also the discussion of the function assignF)

7.3 call release



This schema specifies the release of a call.
An engineer releases a call back to the system by removing his/her name from ownership of the call.
(see also the discussion of the function releaseF)

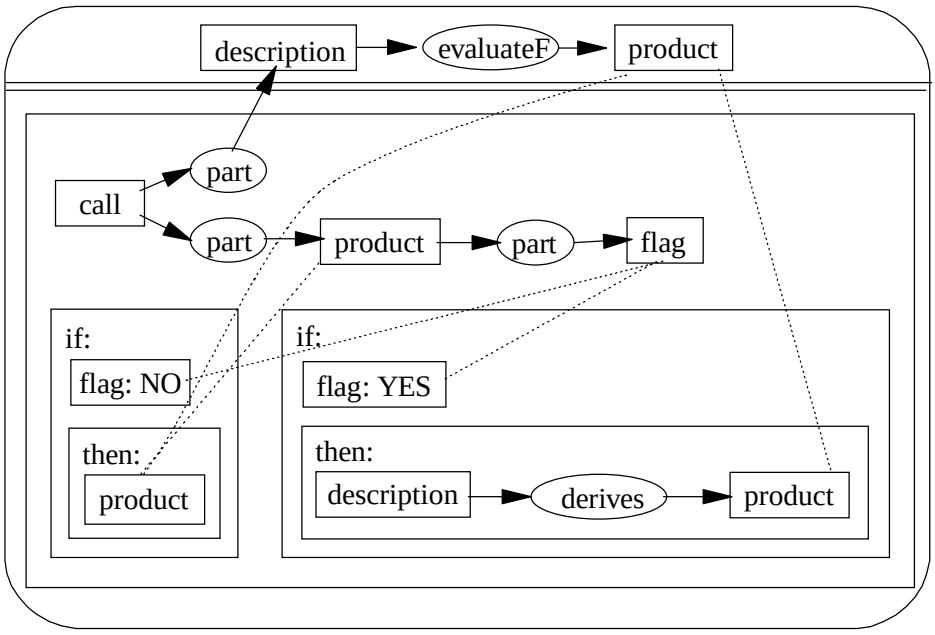
7.4 assignment notification



This schema specifies the notification of a call.
Assignment of a call is announced to the new owner engineer.
(see also the discussion of the function notifyF)

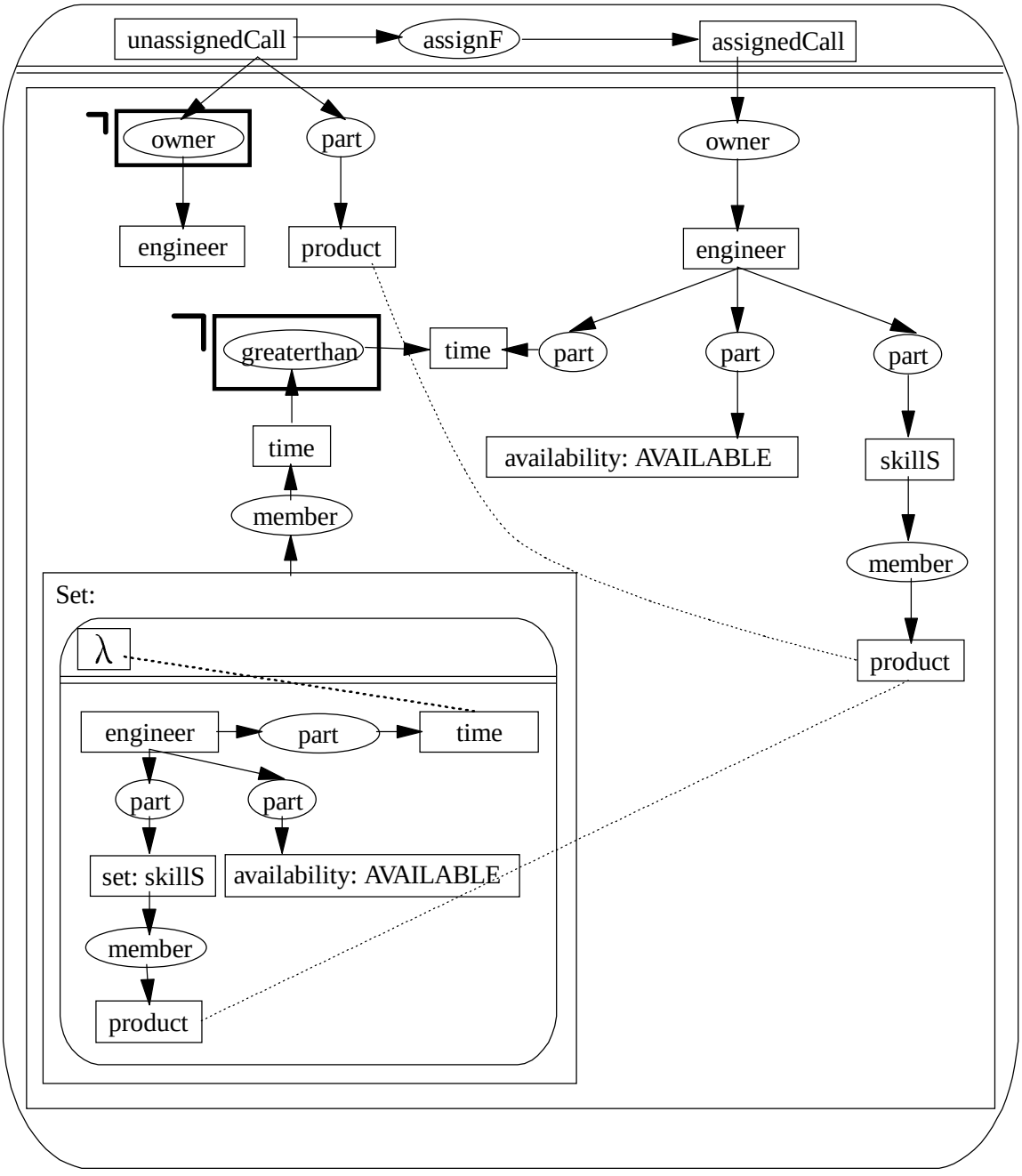
8 Function Specifications

8.1 evaluateF



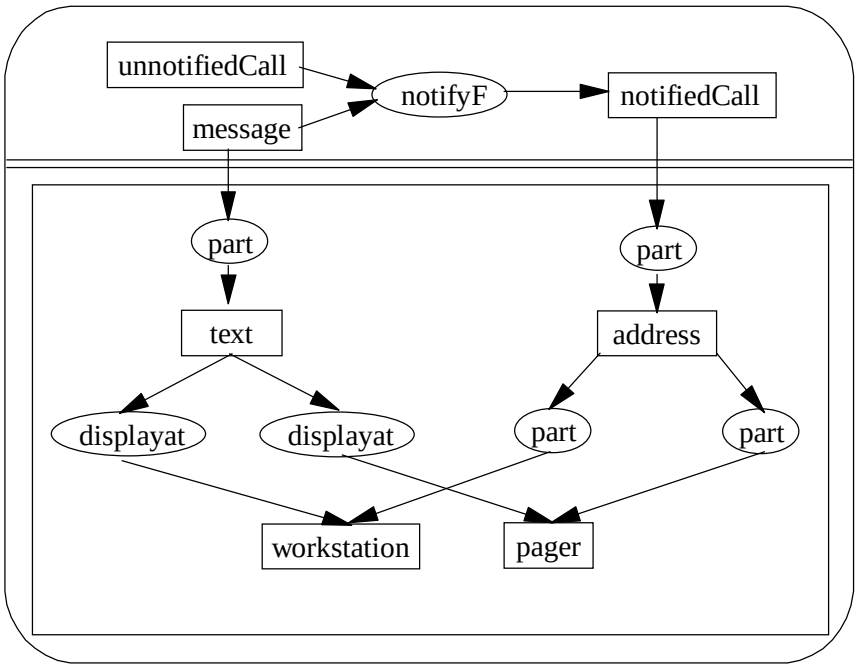
If a call's flag attribute is YES, then the evaluateF function analyses the description associated with the call to derive a revised product. The call's product is replaced by the derived product. If a call's flag attribute is NO, the call's product is left unchanged.

8.2 assignF



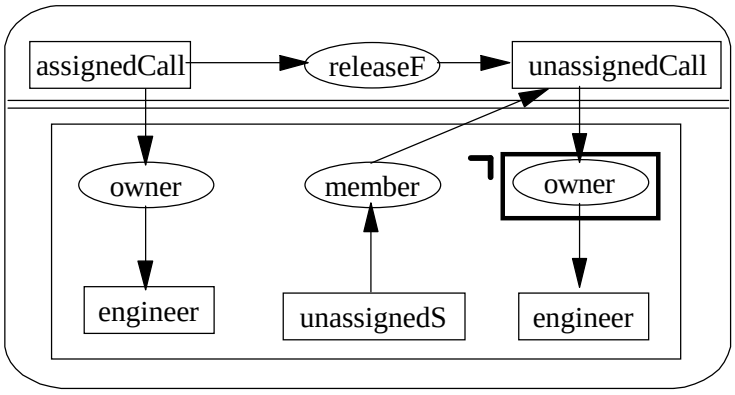
The assignF function takes a call which has no owner and assigns the call to an engineer who has the skill necessary to handle the product associated with the call, is AVAILABLE, and who has not been assigned a call more recently than any other available engineer.

8.3 notifyF



Sends a call assignment message to an engineer's workstation and pager.

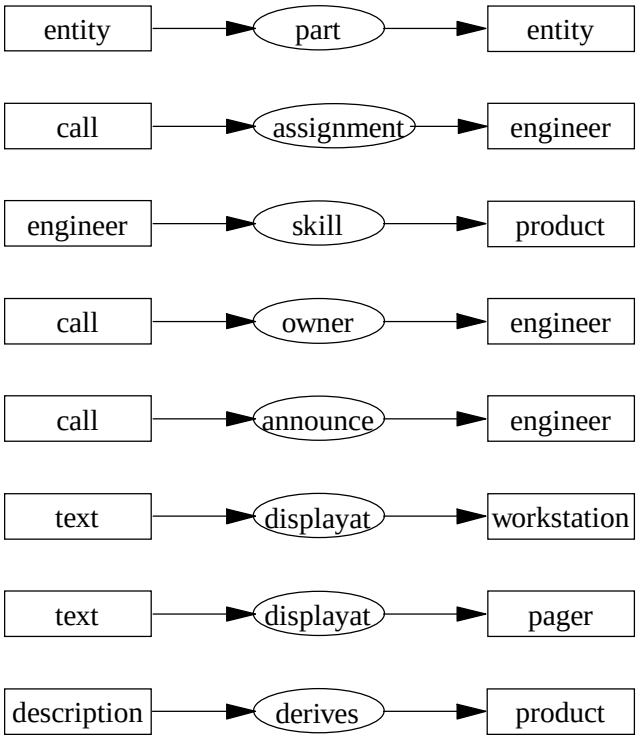
8.4 releaseF



The releaseF function removes ownership of a call from an engineer and returns the call to the unassigned set.

9 Basis Graphs

The relations **domain**, **range**, **member**, **difference**, **union**, **equals**, **greaterthan**, and **next** are assumed to be defined.



10 Observations

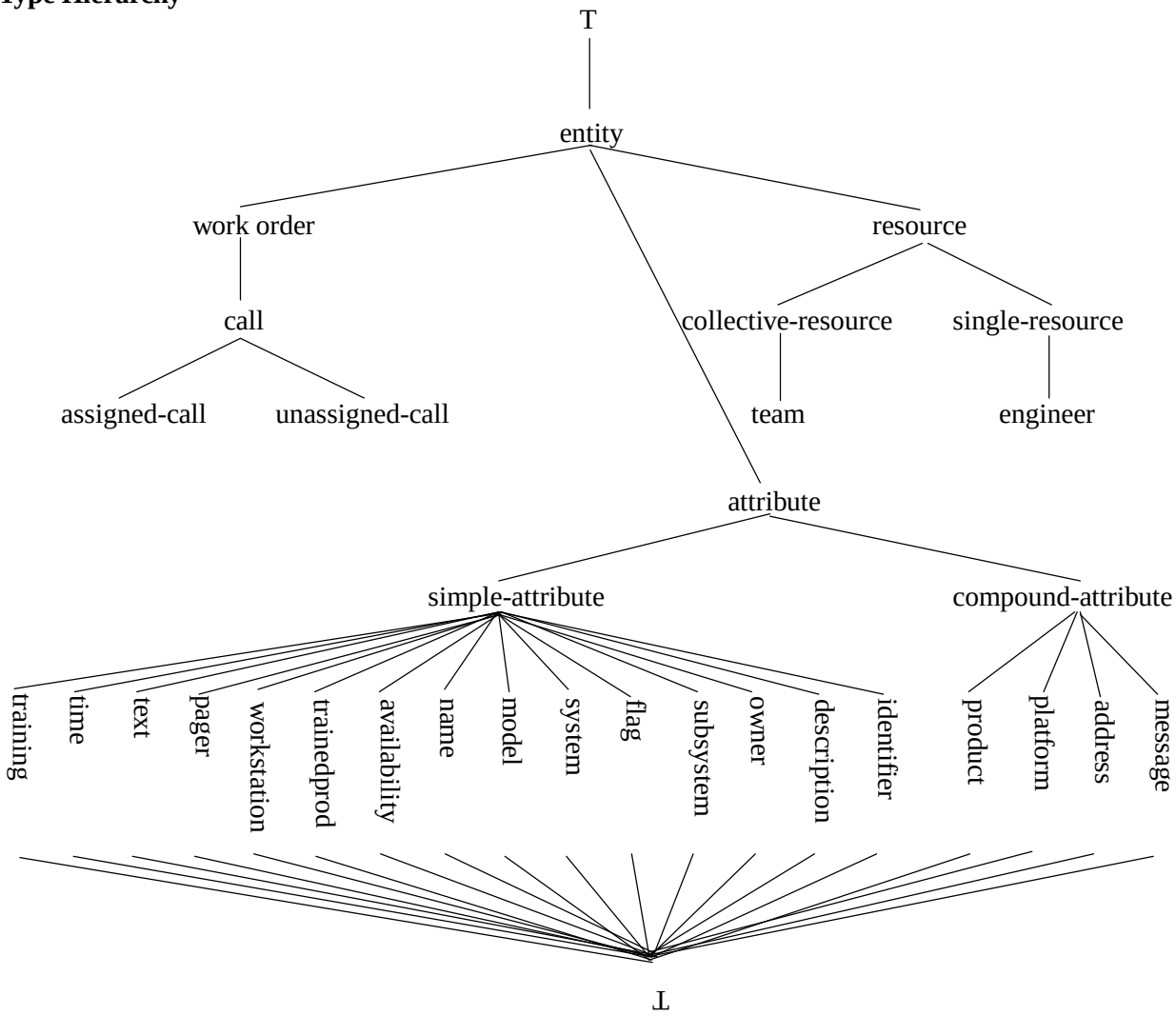
While the model presented here is a simplified model of an existing work order allocation system, this paper does show that conceptual graphs are capable of representing real world, complex systems.

Once defined and drawn, conceptual graphs can convey the overall concept of a system to the reader. I find that conceptual graphs provide a level of abstraction often necessary to take in the overall picture, and suggest that they are most useful in system specification. Defining and drawing the graphs however can be somewhat tedious. A graphical interface to the PEIRCE system may be useful here.

Overall I found that using conceptual graphs to define this system was instructive in the logic of the system. Certainly a system which would convert the graphs to executable code would be extremely useful, especially for rapid prototyping of systems.

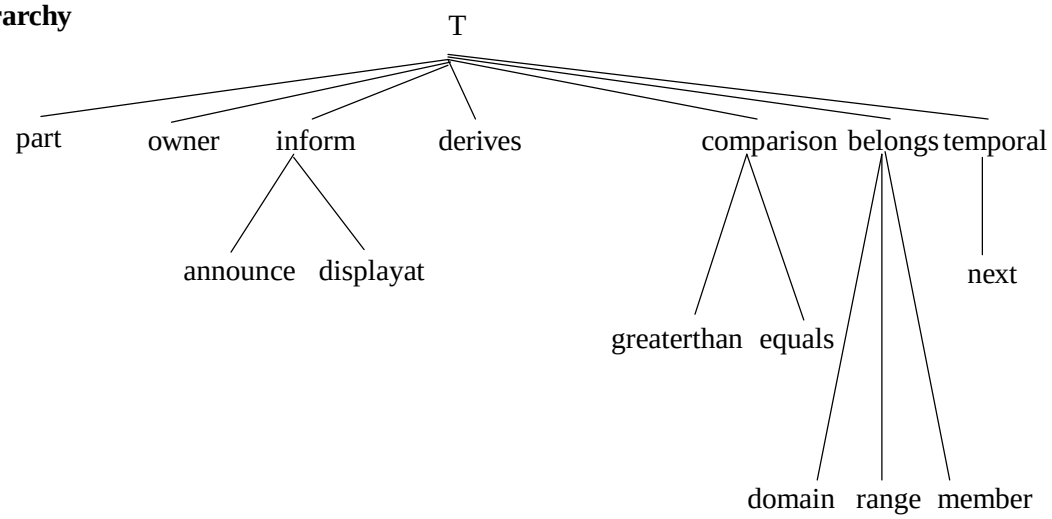
Appendix A

Type Hierarchy



Appendix B

Relation Hierarchy



References

[Ell96] Gerard Ellis. *Lecture notes in Knowledge Representation* - CS433, Department of Computer Science, RMIT, 1996.

[Ell94] Gerard Ellis, editor. *Case Studies in Conceptual Graphs* - Technical Report, August 1994.

[Sow96] John F. Sowa. *Knowledge Representation: Logical Philosophical, and Computational Foundations*. PWS Publishers, 1996.