Understanding Role Concepts
Working Note 20

Peter Clark, John Thompson, Mike Uschold
Knowledge Systems
Boeing Mathematics and Computing Technology
MS 7L66, PO Box 3707, Seattle, WA 98124
{peter.e.clark,john.a.thompson,
michael.f.uschold}@boeing.com

Bruce Porter
Computer Science Dept.
University of Texas
Austin, TX 78712
porter@cs.utexas.edu

November 2000

Abstract

"Role concepts" are an intuitive but problematic notion in knowledge representation. They appear to be intuitive because we often talk about objects "playing a role" in everyday language without controversy, while they are simultaneously problematic because it turns out to be difficult to formalize exactly what this means. This short note attempts to make sense of "role concepts" in both a pragmatic and philosophical way. In the pragmatic part, we work with an intuitive notion of a role as "the way an object participates in a relationship", and describe how roles can be represented in a knowledge-base. We also draw some important distinctions between objects playing a role, objects potentially able to play a role, and objects intended to play a role. In the philosophical part, we argue that this definition of roles does not, in fact, adequately distinguish them from any other type of concept, and instead suggest an alternative meaning in terms of how a theory may change with time. The conclusion of this is that, while it is sometimes reassuring to identify role concepts, their identification should not impact the organization of knowledge in a knowledge-base.

1 Introduction

"Role concepts" are a particularly problematic notion in knowledge representation. Informally, a role might be described as "the way an entity participates in a relationship (e.g., a person may be a customer in a sale)" [1], and thus a role concept is the class of all such participants. Examples might include "teacher", "manager", "producer", and "consumer". People seem to have a strong intuitive notion of what role concepts are, and that they are somehow distinct from other concepts (the philosopher Pierce named these as concepts having "firstness" and "secondness" properties respectively). However, they are also problematic because it turns out to be difficult to pin down this distinctiveness in a formal way. To further add to the confusion, people often conflate the concepts of things which *are* in a role, things which are *potentially able* to fill a role, and things which are *intended* to fill a role, making literature on the topic even more confusing.

This working note has two purposes, one pragmatic and one philosophical. In the pragmatic part, we work the above intuitive definition of a role ("the way an object

participates in a relationship"), and describe how roles can be represented in a knowledge-base. We also draw some important distinctions between objects playing a role, objects potentially able to play a role, and objects intended to play a role. In the philosophical part, we argue that this definition of roles, although useful for the former discussion, does not adequately distinguish them from any other type of concept, and instead suggest an alternative meaning in terms of how a theory may change with time. The potentially controversial conclusion of this paper is that, while it is sometimes reassuring to identify role concepts, their identification should not impact the organization of knowledge in a knowledge-base.

In this note, we provide formal statements in both logic and the representation language KM [2].

This Working Note supercedes two earlier (and in places erroneous) Working Notes on this topic [3, 4].

2 A Pragmatic Treatment of Role Concepts

2.1 What is a Role concept?

Uschold et al. [1] define a role as "the way in which an ENTITY participates in a RELATIONSHIP (e.g., a person may be a customer in a sale)." As mentioned, we will question this definition later in Section 3, but work with it for now. Following this definition, a useful way to think about roles in logic is as labels for the arguments of a predicate. For example, if Sue manages Fred we might assert that

```
;;; "Sue manages Fred"
;;; manages(Sue, Fred)
manages(*Sue,*Fred)                ;; KM notation ('*' denotes instances)
```

and then say Sue is playing the role of "manager", and Fred the role of "employee" – in other words, objects which fill the first argument of (at least one) *manages*() assertion are managers, and objects which fill the second argument of (at least one) *manages*() assertion are employees. If we wanted, we could introduce the role class *Manager* defined exactly in this way:

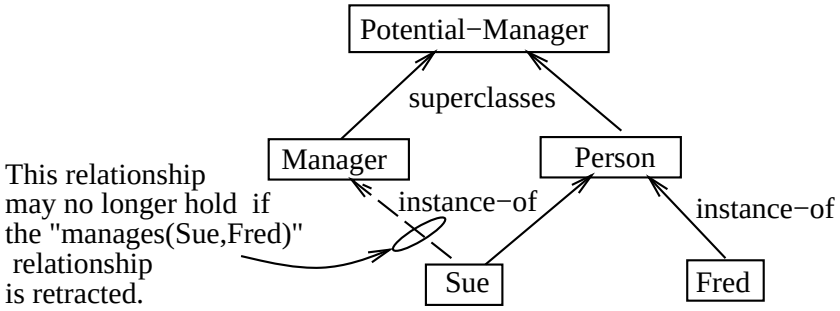
```
;;; "If you manage something, then you are a Manager (and vice versa)."
;;; ∀m (∃e manages(m, e)) ↔ isa(m, Manager)
(every Manager has-definition
  (manages ((a Thing))))
```

Note that if our theory about the world changes, then the instances of the class *Manager* may change, as managers (here) are those things which instantiate the (first argument of) the *manages*() relation in the our current theory. Again, we will delve deeper into this issue of variation with time later in Section 3.

2.2 Actual and Potential Role Concepts

We must be careful to distinguish things which *are* managers from those which *could be* managers, as these are two separate concepts. In the Enterprise Ontology [1], Uschold et al. make this explicit by introducing both concepts and giving them separate names. We will do the same here, using `Manager` to refer to (the class of) things which *are* currently managers, and `Potential-Manager` to refer to things which *could be* managers. If we

are using situation calculus, we may expect instances of the former to change between situations, but members of the latter to remain the same. Also, clearly, every `Manager` is also a `Potential-Manager`. A fragment of the taxonomy from the theory containing $manages(Sue, Fred)$ might look:



Note that in our world (and thus our theory modeling it), we expect the members of the role class `Manager` to change with time (i.e., each day, people start and stop being managers), but members of the "potential role" class `Potential-Manager` to be fixed. Often there is little we can say about these "potential role" concepts, for example almost any object can play the role of a portal covering (lid) of a container, and so the class `Potential-Portal-Covering` (if we were to reify it) would be high in the taxonomy, perhaps immediately above `Tangible-Entity`. These potential role concepts may thus not be that useful in practice, although conceptually it is very useful to bear them in mind to avoid confusing them with their corresponding role concepts.

2.3 Reifying Relationships

A common syntactic trick in knowledge representation, in particular for frame-based systems, is to reify a ground Nary predicate assertion as an instance, and then use a set of binary predicates to relate that instance to its arguments. For example, we can re-express the assertion $manages(Sue, Fred)$ as:

$$;;; \exists x\ isa(x, Managing) \wedge manager(x, Sue) \wedge employee(x, Fred)$$

or $consumes(Cell1, Molecule1, Cytoplasm1)$ as

$$\exists x\ isa(x, Consuming) \wedge agent(x, Cell1) \wedge patient(x, Molecule1) \wedge location(x, Cytoplasm1)$$

or $container(Wall1, 2DSurface1, Door1)$ as

$$\exists x\ isa(x, Container) \wedge has\text{-}wall(x, Wall1) \wedge has\text{-}portal(x, 2DSurface1) \wedge has\text{-}portal\text{-}covering(x, Door1)$$

If we rewrite assertions in this way, roles change from being (metalogical) labels for a predicate's arguments to being binary predicates in the representation language itself. This is convenient as it allows us to refer to roles explicitly by name (using those predicates), rather than by argument position. We will use this convention from now on.

2.4 Giving Semantics to the Relationship

Roles are labels for participants in a relationship. But what gives the meaning (semantics) to that relationship? As for anything else, we can write assertions stating that the existence of that relationship implies various other things. For example:

;;; "For all containers, the walls are tangible entities."
;;; $\forall c, w\ isa(c, Container) \wedge has\text{-}wall(c, w) \rightarrow isa(w, Tangible\text{-}Entity)$

```
(every Container has
  (has-wall ((a Tangible-Entity with
               (intersects ((the has-portal of Self))))))
```

;;; "For all containers, the portal intersects the wall." (assuming exactly 1 wall)
;;; $\forall c, w, p\ isa(c, Container) \wedge has\text{-}wall(c, w) \wedge has\text{-}portal(c, p) \rightarrow intersects(w, p)$

```
(every Container has
  (has-portal ((must-be-a 2D-Surface with
               (intersects ((the has-wall of Self)))))))
```

2.5 Do we really need Role Classes?

Earlier, we carefully distinguished the class of things currently playing a role, from those which *potentially* could play a role. Do we really need the former?

Suppose we want to represent the statement "All employees are happy." We have two stylistic alternatives, which we show in both logic and KM notation:

1. (a) First define the role class `Employee`, either by asserting:

   ;;; "All things in the `employee` role of a `Manages` are `Employee`s."
   ;;; $\forall x, y\ isa(x, Manages) \wedge employee(x, y) \rightarrow isa(y, Employee)$
   ;;; [or equivalently:
   ;;; $\forall y\ isa(y, Employee) \leftarrow \exists x\ isa(x, Manages) \wedge employee(x, y)$[1]]

   ```
   (every Manages has
     (employee ((must-be-a Employee))))
   ```

   or by asserting the stronger statement:

   ;;; "All things in the `employee` role of a `Manages` are `Employee`s,
   ;;; and vice versa."
   ;;; $\forall y\ isa(y, Employee) \leftrightarrow \exists x\ isa(x, Manages) \wedge employee(x, y)$

   ```
   (every Employee has-definition
     (employee-of ((a Manages))))
   ```

   (b) then make the statement about members of that reified class:

   ;;; "All employees are happy."
   ;;; $\forall x\ isa(x, Employee) \rightarrow mood(x, Happy)$

   ```
   (every Employee has
     (mood (*Happy)))
   ```

2. Don't reify the class `Employee`, and instead express the statement as a direct implication for every `Manages` instance:

   ;;; "All employees (i.e., things in the `employee` role of a `Manages`) are happy."
   ;;; $\forall x, y\ isa(x, Manages) \wedge employee(x, y) \rightarrow moood(y, Happy)$

   ```
   (every Manages has
     (employee ((must-be-a Person with
                  (mood (*Happy))))))
   ```

These two alternative approaches are logically equivalent, the difference being partly stylistic and partly inferential. In KM, the first approach is more verbose, computationally more costly, and adds a level of indirection between the `Manages` and `Employee` concepts.

[1] as $\forall x, y\ f(x) \rightarrow g(y) \equiv \forall y\ [\exists x\ f(x)] \rightarrow g(y)$

Conversely, the second loads all assertions related to `Manages` on a single frame, making it a potentially cumbersome structure if there were many such assertions. The appropriate style to use will depend on the details and complexity of the particular concept being encoded. To answer the original question in this subsection's title, namely "Do we really need role classes?", the answer is "No, we don't *have* to reify role classes, but sometimes it is conveniently to do so."

2.6 Do we really need "Potential Role" Classes?

As a separate issue, we can also write axioms using "potential role" classes (i.e., the class of things which can potentially fill a particular role), for example:

```
;;; "Portal-Coverings must be of type Potential-Portal-Covering."
;;; ∀x, y isa(x, Container) ∧ has-portal-covering(x, y)
;;;                  → isa(y, Potential-Portal-Covering)
(every Container has
  (has-portal-covering ((must-be-a Potential-Portal-Covering))))
```

This enforces a useful type constraint on (actual) portal-coverings of containers. Note it is not strictly necessary to reify the class `Potential-Portal-Covering`, although sometimes it might be convenient, e.g., if several classes of objects were considered types of potential portal coverings, and we often wanted to refer to a member of one of those classes. Again, the choice of reifying the concept `Potential-Portal-Covering` is a matter of stylistic convenience.

2.7 Relationships vs. Roles in Relationships

Another important distinction to make is between a *relationship* (e.g., *manages*(), *container*()), and *roles in* that relationship (eg. manager, employee, portal, portal-covering). In particular, the relation *container*() (or equivalently its reified class `Container`) is *not* a role in a relationship, but a relationship itself (between a wall, a cavity, and some surfaces). Thus, by our earlier definition, `Container` is not a role concept, while `Portal`, `Portal-Covering` etc. are (NB not `Potential-Portal`, `Potential-Portal-Covering`).

So what do we mean when we say "X can play the role of a container," if container is not a role concept? We have to be careful here, as there are two interpretations of this sentence. The first is that we really mean "X can be *viewed as* a container," meaning that although X is not declared as a container in the KB, we can treat it as a container for the purpose of a particular task. This in turn can mean treating X literally as a container (temporarily assert that "X isa container" for the duration of that task), or treating it metaphorically as a container (in which case only some of the container axioms, possibly with modification, are applied to X). This notion of views is interesting and important in a knowledge-base, but is orthogonal to the issue of role concepts. It is discussed in a separate Working Note [4].

The second interpretation of "X can play the role of a container," is that the word "container" here denotes a *role* in a *different* relationship, rather than the concept of a container itself. Under this interpretation, the sentence is incomplete, missing the completing phrase "in a ... relationship". For example, we might consider a board-game to comprise a board, some pieces, some dice, and a container to shake the dice in, and then say (when we can't find the original container) "My cup can play the role of the container for this board-game." Here "container" refers to the container *role* within the *board-game* system of relationships (namely the 4th argument of *board-game*(*board*, *pices*, *dice*, *cup*)),

and this is distinct from the container *relationship* between a wall, a cavity, and some surfaces ($container(wall, cavity, surfaces)$). We might call the "container" role in the board-game the "container-for-board-game" role, and the class of (actual) fillers for it `Containers-Currently-Being-Used-In-Board-Games`, to avoid confusion with the earlier class `Container`, denoting the (class of) sets of objects related in a containerhood way.

2.8 Intended Purpose

What is the relationship between the class `Door`, the role class `Portal-Covering` (objects which are actually assigned to cover a container's portal), and the class `Potential-Portal-Covering`? Clearly, `Door` is not a subclass of `Portal-Covering` (e.g., doors for sale in the hardware store are not yet assigned to any container) but is a subclass of `Potential-Portal-Covering` (doors certainly *can* be portal coverings). But this doesn't quite tell the whole story – we could say the same about the concept `Book`, for example (I can make a book a `Portal-Covering` by placing it on my cup), but intuitively books and doors should have some different relationship to portal covering. What's missing here is a third notion, namely *intended purpose*: The intended purpose of a door is to be used as a portal covering. We could introduce a third concept, `Intended-Portal-Covering` to represent objects which were intended (by their maker) to be portal coverings. (Presumably `Intended-Portal-Covering` would normally be a subclass of `Potential-Portal-Covering`, but we can imagine exceptions to this.) Again, conceptually, it is useful to be aware of this third class of objects, although in practice it is probably not useful to explicitly introduce it into a KB. Rather, one could make "intended purpose" assertions on each separate class of object intended to be portal coverings (e.g., `Door`), thus achieving the same effect. We do not discuss how to represent intended purpose here, as this is not the topic of this paper.

2.9 Correlated Type Restrictions

Finally, let us return to the classes of "potential" role concepts, such as `Potential-Portal-Covering` discussed earlier. We would like to represent the fact that, as we consider more specific types of containers, the type restrictions on those containers' roles also become more specific. For example, a room's walls must be of type brick (say), and its portal covering must be a door; a tea-pot's walls must be made of china (say), and its portal covering must be a lid; etc. Note that the type restrictions on roles are correlated: although lids and doors are both portal coverings, the system should complain at a lid covering a doorway, or a door covering the body of a teapot.

These correlated type restrictions are nothing mysterious, and can be implemented simply by defining the appropriate, specialized type of containers. For example:

```
;;; "All container walls of rooms are made out of brick."
;;; ∀r, w isa(r, Room) ∧ has-container-wall(r, w) → made-of(w, Brick)
(every Room has (has-container-wall ((must-be-a Wall with (made-of (*Brick)))))

;;; "All portal coverings of rooms must be doors." (here ignoring windows etc.)
;;; ∀r, p isa(r, Room) ∧ has-portal-covering(r, p) → isa(p, Door)
(every Room has (has-portal-covering ((must-be-a Door))))
```

```
;;; ———-
;;; "All container walls of tea-pots are made out of china."
;;; ∀t, w isa(t, Tea-Pot) ∧ has-container-wall(t, w) → made-of(w, China)
(every Tea-Pot has (has-container-wall ((must-be-a Wall with (made-of (*China)))))

;;; "All portal coverings of tea-pots are lids."
;;; ∀t, p isa(t, Tea-Pot) ∧ has-portal-covering(t, p) → isa(p, Lid)
(every Tea-Pot has (has-portal-covering ((must-be-a Lid))))
```

3 A Re-Characterization of Role Concepts

We have given two independent characterizations of role concepts, namely:

- A class of objects defined by their participation (in a particular way) in a particular relationship. For example, `Manager` is a role concept because it denotes the class of things filling the first argument of (at least one) $manages(x, y)$ assertion.
- Classes whose members may change with time. `Manager` is a role concept because people are not intrinsically managers or non-managers – rather, people become and cease to be managers depending on the activities in which they are currently engaged.

We now probe a bit deeper into these.

3.1 Roles as Participants in a Relationship

We make the observation that concepts "defined by their [members'] participation in a particular relationship" are also known simply as "defined concepts" in the literature, a topic which has been studied by the Description Logic community for many years. The concept `Red-Wine`, for example, is defined by its simultaneous participation in the "color" relationship to `Red`, and the "superclasses" relationship to `Wine`, although intuitively we would not consider `Red-Wine` to be a role. (Or, if the reader objects to relationships with instances like `Red`, consider `European-Wine` or `Manufactured-Wine`). Thus the notion of necessary participation in a relationship does not seem to hold up as a distinguishing characteristic of role concepts. In fact, as Pat Hayes has pointed out [5], pretty much everything necessarily participates in *some* relationship (e.g., people necessarily have a time of birth), and extensive attempts to finesse this definition to (for example) particular types of relationship have not succeeded (e.g., see [6, Search for "Roles"]).

This does not necessarily mean that the notion of role concepts is meaningless – rather, it merely means that the fact that role concepts are also defined concepts is not their distinguishing feature, and so we must look elsewhere. In fact, the changability of their class members provides such a distinguishing feature, as we now discuss.

3.2 Roles as Concepts with Changing Members

Throughout this paper we have also referred to role concepts as ones whose members "may change with time". But what exactly does this mean? A first-order theory itself, including statements about which objects are in which classes, never "changes with time", it just is. Of course, we could add or delete an axiom, but then it would be a different theory, and so our notion of "changing with time" is really an extra-logical reference to things we might do to a theory. Sue might cease to be a manager if we retract $manages(Sue, Joe)$, but

then again Sue might cease to be a person if we retract or modify part of the definition of *Person*. What counts as a valid modification to a theory? Presumably our theory should reflect the world, and so valid changes to the theory are those reflecting possible changes in the world; and so statements about which concepts have a fixed, permanent set of members are really statements about how (we believe) the world might change.

We can clarify this definition by saying that role concepts are those whose members may change with time "in the real world." Although this ties role concepts to our subjective opinion about what we consider possible in the world, it seems about the best we can do to formalize our intuitive notion of "role".

Perhaps more importantly, this definition does not impact a static, first-order theory about the world at a moment in time. That is, given a first-order theory describing a "snapshot" of the world, the labeling of which concepts are role concepts is irrelevant, as the semantics of role concepts concerns how theories may change, not how they actually are at any particular moment. A statement about which concepts are roles is a statement about the *set* of possible theories which (we consider) are representations of the way the world can be, rather than a statement about any particular theory within that set. Or, in a situation calculus framework, a statement about roles is a statement about the *set* of situations we consider possible/allowable (through inference or user-action, depending on what we admit as "possible"), rather than a statement about any particular situation. An axiomatization of non-role concepts would state that for all non-role concepts, the set of concept members is the same in all possible situations. Role concepts would then be those for which this constraint did not hold. Similarly in a non-situation calculus framework, where we have a "current theory" representing the current state of the world which we then manually/automatically modify to represent change, identifying role concepts would be irrelevant to the theory itself, and instead would only impact what modifications were considered allowable (which is outside the logic framework of the theory itself).

4 Summary and Conclusion

This note is perhaps a mixture of clarification and confusion. In terms of clarification, we have highlighted important distinctions between objects *playing* a role, *capable of playing* a role, and *intended to play* a role. We have also illustrated that whether role concepts should be reified in the knowledge-base is a pragmatic question of style, rather than one with a definitive answer.

In terms of confusion, throughout the paper we have used an intuitive meaning of roles as "the way an object participates in a relationship," only to knock it down at the end as a true but undistinguishing characteristic. We then argued that a more meaningful definition is in terms of the ways in which a theory might change. This approach was deliberate, as we wanted to provide intuitive comments on handling roles, even if (as it turns out) those comments are equally valid for any defined concept. The bottom line is that, while it may sometimes be helpful to think of some concepts as "role concepts", the meaning of this label should not impact the organization of knowledge in a knowledge-base.

References

[1] Michael Uschold, Martin King, Stuart Moralee, and Yannis Zorgios. The enterprise ontology. *Knowledge Engineering Review*, 13(1):31–90, Mar 1998.

[2] Peter Clark and Bruce Porter. KM – the knowledge machine: Users manual. Technical report, AI Lab, Univ Texas at Austin, 1999. (http://www.cs.utexas.edu/users/mfkb/km.html).

[3] Peter Clark and Bruce Porter. Working note 11: Should role concepts be included in a taxonomy? (http://www.cs.utexas.edu/users/pclark/working_notes), 1998.

[4] Peter Clark, John Thompson, and Bruce Porter. Working note 19: Handling role concepts and views in a knowledge-base. (http://www.cs.utexas.edu/users/pclark/working_notes), 2000.

[5] Pat Hayes. Re:roles, again. http://ksl-web.stanford.edu/email-archives/srkb.messages/583.html (Discussion on the SRKB Mailing List), 1995.

[6] SRKB List. (discussion on the srkb mailing list). http://ksl-web.stanford.edu/email-archives/srkb.index.html, 1995.