Inference Graphs: A Roadmap

Daniel R. Schlegel DRSCHLEG@BUFFALO.EDU
Stuart C. Shapiro SHAPIRO@BUFFALO.EDU
Department of Computer Science and Engineering, University at Buffalo, Buffalo, NY 14260 USA

Abstract

Logical inference is one approach to implementing the reasoning component of a cognitive system. Inference graphs are a method for natural deduction inference which, uniquely in logic-based cognitive systems, use concurrency to reason about multiple possible ways to solve a problem simultaneously, and cancel no-longer-necessary inference operations. We outline extensions to inference graphs which increase their usefulness in cognitive systems, including: the use of a more expressive logic; a method for "wh- question" answering; and a way to focus reasoning on problems which cannot immediately be answered due to incomplete information, so when more information becomes available the inference can proceed. We discuss how these three improvements increase the usefulness of inference graphs in cognitive systems.

1. Introduction

Logical inference is one method of implementing the reasoning component of a cognitive system. Examples include GLAIR (Shapiro & Bona, 2010) and MGLAIR (Bona, 2013), which use the SNePS 2 KRR system (Shapiro & The SNePS Implementation Group, 2010) as their knowledge layer. SNePS 2 uses a first order logic, and can perform natural deduction inference using forward, backward, bi-directional (Shapiro, Martins, & McKay, 1982), and a limited form of focused reasoning. Another example is NARS (Wang, 2006), which reasons using deductive, inductive, and abductive reasoning on terms which have degrees of truth based on evidence.

The latest member of the SNePS family – CSNePS – uses a new graph-based natural deduction reasoning system known as *Inference Graphs* (Schlegel & Shapiro, 2013a). Inference graphs are designed to support forward, backward, bi-directional, and focused reasoning. Thus far, inference graphs have been implemented for propositional logic using concurrent processing techniques – a unique feature in logic-based cognitive systems. Concurrency allows inference graphs to reason about multiple possible ways to solve a problem simultaneously, and notice (and cancel) reasoning operations which are no longer necessary due to derivations which have already taken place.

Propositional logic is insufficient for complex conceptual reasoning because it is unable to represent generic concepts, for example, "*all Dobermans*", (as opposed to specific concepts, for example, "*Fido*") which humans deal with every day. We propose the use of a first order logic known as $\mathcal{L}_A$ (Shapiro, 2004) – the logic of arbitrary and indefinite objects. An advantage of $\mathcal{L}_A$ is that it makes reasoning about generic concepts easy. $\mathcal{L}_A$ also makes some types of subsumption reasoning natural.

In this paper, we will discuss extensions to inference graphs which will make them a more powerful tool for inference, allowing them to solve more problems, and thus increasing their usefulness in modeling human cognition. While much of the work discussed is not yet implemented in CSNePS, we will motivate the work by discussing the advantages such additions will provide towards the goal of human-level AI, and justify our choices as compared to competing approaches.

In Section 3 we provide an introduction to the current implementation of inference graphs for propositional logic and the relevant parts of CSNePS. Sections 4–6 discuss the extensions to inference graphs we propose, including the use of a more expressive logic (Section 4), question answering (Section 5), and focused reasoning (Section 6). We conclude with a discussion of the expected capabilities of the completed system in Section 7.

2. KRR Requirements for Cognitive Systems

There are many systems of logic. What they have in common are: having a syntax, a formal grammar specifying the well-formed expressions; a semantics, a formal means of assigning meaning to the well-formed expressions; and a proof theory, specifying a mechanism for deriving from a set of well-formed expressions additional well-formed expressions preserving some property of the original set, often called "truth." To the extent that a cognitive agent has a set of beliefs, that we can represent these beliefs in a "language of thought" with a well-defined syntax and semantics, and that the agent's reasoning consists of deriving new beliefs from its old beliefs preserving some notion of rationality, we can view an agent's beliefs and reasoning system to be some logic, though not necessarily standard, classical logic.

Many logic-based reasoning systems have been developed, varying along the dimensions of expressiveness and reasoning style. Along the dimension of expressiveness, families of logical languages that have been used include: Propositional Logic; Finite-Domain Predicate Logic; First-Order Predicate Logic (FOPL); Description Logic; and Horn-Clause Logic. Among these, FOPL is the most expressive. The others have reduced expressiveness, often motivated by issues of tractability (Brachman & Levesque, 1987). Our belief is that cognitive agents that possess human-level intelligence, and that can interact with humans in natural language, must be able to represent their beliefs in a formal language at least as expressive as FOPL (*see* (Iwańska & Shapiro, 2000)).

Along the dimension of reasoning style, basic approaches include: model finding; refutation resolution; refutation semantic tableaux; and proof-theoretic derivation. The approach of model finding is: given a set of beliefs taken to be true, find truth-value assignments of the atomic beliefs that satisfy the given set. The approach of the refutation methods is: given a set of beliefs and a conjecture, show that the set logically entails the conjecture by showing that there is no model that simultaneously satisfies both the given set and the negation of the conjecture. The approach of proof-theoretic derivation is: given a set of beliefs, and using a set of rules of inference from the proof theory, either derive new beliefs from the given ones (forward reasoning) or determine whether a conjecture can be derived from the given set (backward reasoning). We favor proof-theoretic derivation, because it is the only one of these approaches in which an agent derives new beliefs that can be added to its initial set of beliefs, which humans do when they reason.

Several cognitive systems are based on production systems. Production system rules, being pattern-action rules, have more the flavor of programming language statements than beliefs with declarative semantics, and are not typically included in the set of the agent's beliefs. For this reason, we prefer a more logic-based approach.

We recognize the importance of graded levels of truth (or belief), such as provided by credibility or probability theory, to cognitive agents. It is just a matter of research agenda that we do not (yet) take this into account in our cognitive system.

3. Background

3.1 CSNePS

CSNePS is a knowledge representation and reasoning system which is currently being implemented according to the specification of SNePS 3 (Shapiro, 2000). At its core is a knowledge representation scheme which can be seen as simultaneously logic-based, frame-based, and graph-based (Schlegel & Shapiro, 2012). The graph-based representation, with which we are most concerned here, is called a propositional graph.

In the tradition of the SNePS family (Shapiro & Rapaport, 1992), propositional graphs are graphs in which every well-formed expression in the knowledge base, including individual constants, functional terms, atomic formulas, or non-atomic formulas (which we will refer to as "rules"), is represented by a node in the graph. A rule is represented in the graph as a node for the rule itself (henceforth, a *rule node*), nodes for the argument formulas, and arcs emanating from the rule node, terminating at the argument nodes. Arcs are labeled with an indication of the role (*e.g.,* antecedent or consequent) the argument plays in the rule, itself. Every node is labeled with an identifier. Nodes representing individual constants, proposition symbols, function symbols, or relation symbols are labeled with the symbol itself. Nodes representing functional terms or non-atomic formulas are labeled `wft`*`i`*, for some integer, *`i`*. Every SNePS expression is a term, denoting a mental entity, hence `wft` instead of `wff`. An exclamation mark, "`!`", is appended to the label if the proposition is asserted in the KB. No two nodes represent syntactically identical expressions; rather, if there are multiple occurrences of one subexpression in one or more other expressions, the same node is used in all cases. Propositional graphs are built incrementally as terms are added to the knowledge base, which can happen at any time.

Each term in the knowledge base has a semantic type, itself existing within an ontology of semantic types which the user can add to (see Figure 1). Parent types are inherited by instances of child types, and sibling types are mutually disjoint, but not exhaustive of their parent. All terms are descendants of the type Entity. Objects in the domain should be an instance of Thing. The types Act, Policy, and Action are not yet used, but will be part of the CSNePS acting system once it has been developed, and allow integration with the MGLAIR cognitive architecture. Terms with the type Propositional are those used to express Propositions and "wh-" style queries (WhQuestion, see Section 5). Only Propositions may be asserted (taken to be true) in the knowledge base.

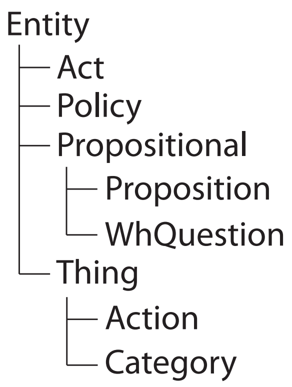


Figure 1: The default CSNePS Semantic Type ontology.

3.2 Inference Graphs for Propositional Logic[1]

An inference graph is a propositional graph in which certain arcs and certain reverse arcs are augmented with *channels* through which information can flow – meaning the inference graph is both a representation of knowledge and the method for performing inference upon it. Channels come in two forms. The first type, *i-channels*, are added to the reverse antecedent arcs – named as such since they carry messages reporting that "I am true" or "I am negated" from the antecedent node to the rule node. Channels are also added to the consequent arcs, called *u-channels*, since they carry messages to the consequents which report that "you are true" or "you are negated." Rules are connected by shared subexpressions. Channels are added as described whenever a rule is added to the graph.

Each channel contains a valve. Valves allow or prevent the flow of messages forward through the graph's channels. When a valve is closed, any new messages which arrive at it are added to a waiting set. When a valve opens, messages waiting behind it are sent through.

Messages of several types are transmitted through the inference graph's channels, serving two purposes: relaying newly derived information, and controlling the inference process. A message can be sent to relay the information that its origin has been asserted or negated (an `i-infer` message), that its destination should now be asserted or negated (`u-infer`), or that its origin has either just become unasserted or is no longer sufficiently supported (`unassert`). These messages flow forward through the graph. Other messages flow backward, controlling inference by affecting the valves: `backward-infer` messages open them, and `cancel-infer` messages close them. The use of messages to control valves allows inference graphs to perform forward, backward, bi-directional, and focused inference.

Inference operations take place in the rule nodes. When a message arrives at a rule node the message is translated into Rule Use Information, or RUI (Choi & Shapiro, 1992). RUIs contain information about how many (and specifically which) antecedents of a rule are known to be true or false, along with a set of support. All RUIs created at a node are cached. When a new one is made, it is combined with any already existing ones. The output of the combination process is a set of

1. The material in this section is mostly taken from (Schlegel & Shapiro, 2013a; Schlegel & Shapiro, 2013b).

new RUIs created since the message arrived at the node. By examining the number of known true or false antecedents, this set is used to determine if the rule node's inference rules can be applied. RUIs prevent re-derivations and cut cycles of message-flow in the graph by ignoring arriving RUIs already in the cache. The disadvantage of using natural deduction is that some rules are difficult to implement such as negation introduction and proof by cases. For us, the advantages in capability and in human users' ability to understand the derivations outweigh the difficulties of implementation.

Once a term is derived or a rule fires, inference operations in additional rules still attempting to derive the term, or cause the rule to fire may no longer be necessary. When this is recognized, `cancel-infer` messages can be sent recursively backward through the graph to halt unnecessary inference, allowing for "eager-beaver" inference, where multiple inference paths are tried simultaneously, and after one succeeds, the others are canceled. All messages are prioritized so inference operations are performed efficiently.

To illustrate these inference mechanisms, Figure 2 shows the process of deriving `f` for the assertions that if `a`, `b`, and `c` are true, then `d` is true, and if `d` or `e` are true, then `f` is true. In this example we use two different, but related connectives – and-entailment and or-entailment. And-entailment requires all of its antecedents to be true, while or-entailment requires only one to be true. In the graph, we prefix the antecedent arcs (labeled `ant`) with $\wedge$ or $\vee$ to make it clear which is in use. Shown as dashed lines are i-channels, while u-channels are shown as dotted lines. In this example, we assume backward inference has been initiated, opening all the valves in the graph. First, in Figure 2a, messages about the truth of `a`, `b`, and `c` flow through i-channels to `wft1`. Since `wft1` is and-entailment, each of its antecedents must be true for it to fire. Since they are, in Figure 2b the message that `d` is true flows through `wft1`'s u-channel. `d` becomes asserted and reports its new status through its i-channel (Figure 2c). In Figure 2d, `wft2` receives this information, and since it is an or-entailment rule and requires only a single antecedent to be true for it to fire, it reports to its consequents that they are now true, and cancels inference in `e`. Finally, in Figure 2e, `f` is asserted, and inference is complete.

4. Reasoning with $\mathcal{L}_A$

The logic of arbitrary and indefinite objects was designed for use as the logic of a KR system for natural language understanding, and for commonsense reasoning (Shapiro, 2004) – goals shared by CSNePS. Several features of $\mathcal{L}_A$ promote these goals, including ease of translation to the logic from natural language by maintaining the locality of natural language phrases, through sharing structure which occurs multiply throughout a text, through a uniform syntax for differently quantified statements, and supporting subsumption inference.

The major functional difference between $\mathcal{L}_A$ and classic FOPL is that $\mathcal{L}_A$ deals with arbitrary and indefinite terms (collectively, quantified terms) instead of universally and existentially quantified variables. That is, instead of reasoning about *all* members of a class, $\mathcal{L}_A$ reasons about a *single* arbitrary member of a class. For indefinite members, it need not be known *which* member is being reasoned about, the indefinite member itself can be reasoned about. Indefinite individuals are essentially Skolem functions, replacing FOPL's existential quantifier. One of the effects of using arbitrary and indefinite terms is that quantified terms may be the result of inference. For example, in

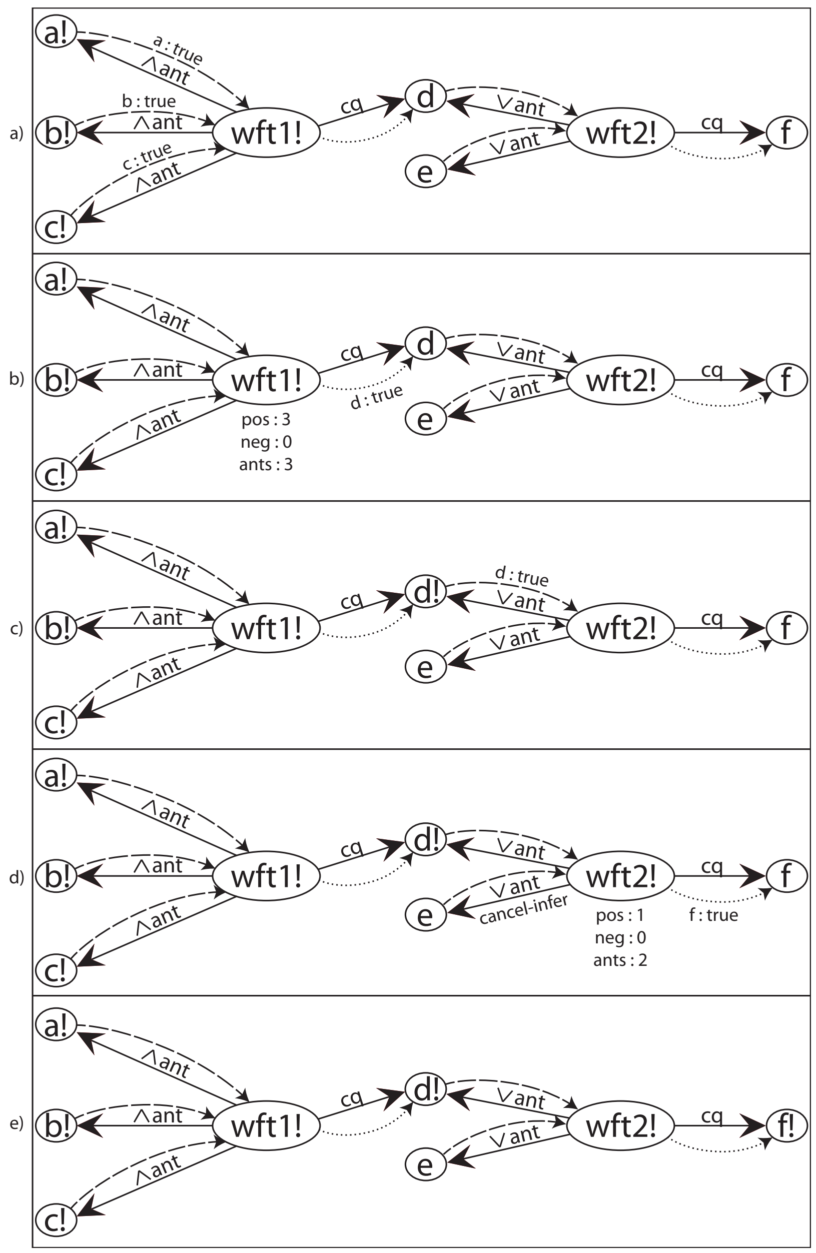


Figure 2: a) Messages are passed from `a`, `b`, and `c` to `wft1`. b) `wft1` combines the messages from `a`, `b`, and `c` to find that it has 3 positive antecedents, of a total of 3. The and-entailment can fire, so it sends a message through its u-channel informing its consequent, `d`, that it has been derived. c) `d` receives the message that it is asserted and sends messages through its i-channel. d) `wft2` receives the message that `d` is asserted. Only one true antecedent is necessary for or-entailment elimination, so it sends a message through its u-channels that its consequent, `f`, is now derived. It also cancels any inference in its other antecedents by sending a `cancel-infer` message to `e`. e) `f` is derived.

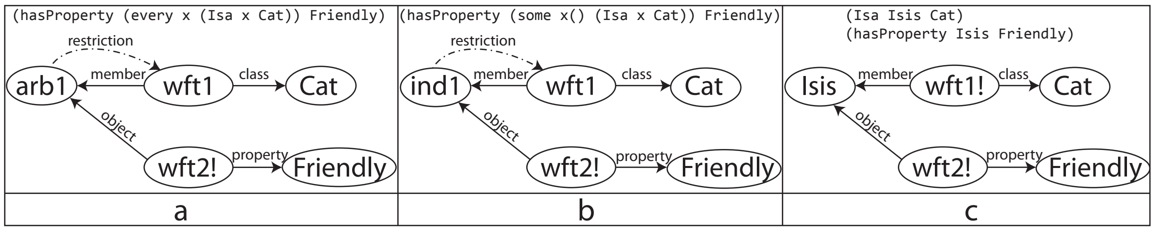


Figure 3: Three graphs with their logical representations illustrating the structural similarities between arbitrary (a), indefinite (b), and ground terms (c).

deriving all dogs known to the system, one answer might be the arbitrary Doberman. This arbitrary term stands for any single Doberman, and may itself have properties shared by all Dobermans.

To our knowledge, the only implemented system which uses a form of arbitrary term is ANALOG (Ali & Shapiro, 1993), though arbitrary objects themselves were first conceived of by Frege, and most famously defended by Fine (Fine, 1983). The logic of $\mathcal{L}_A$ is based on those developed by Ali and Fine (Fine, 1985a; Fine, 1985b), but is different – notably it is more expressive than ANALOG, and designed with computation in mind, unlike Fine's work which omits key algorithms.

A term which contains open occurrences of quantified terms is called a *generic term*. Generic terms and ground terms can be reasoned about similarly. A direct result of the uniform syntax for differently quantified terms used in $\mathcal{L}_A$ is that the graph view of those terms in the KB are also similar. In Figure 3 three different graphs along with their logical equivalents are presented. In Figure 3a, the assertion states that every member of the class Cat has the property of being Friendly. In the graph, an arbitrary term, `arb1`, stands for the arbitrary Cat (because of the restriction arc). Additionally, this arbitrary term has the property of being Friendly. You'll notice in Figure 3b, which is meant to mean that some member of the class Cat is Friendly, that the graph is nearly identical. The only change is that an indefinite term, `ind1`, has replaced `arb1`. The third graph, in Figure 3c, is again very similar to the first two. This is meant to represent the ground assertions that Isis is a member of the class Cat, and that Isis is Friendly.

In modifying the inference graphs to support $\mathcal{L}_A$, we concern ourselves mostly with the deductive inference components, as they are the ones the inference graph is designed to assist with. That said, it is difficult to ignore subsumption inference in $\mathcal{L}_A$. For this reason, we discuss a simplified form of structural subsumption (Woods, 1991) – specifically between quantified terms where no inference is necessary to test subsumption. Related to this, we also discuss simple instantiation of quantified terms where no inference is necessary.[2]

Changes to the inference graph to support inference using quantified terms are concentrated in two areas: the channels, and the rule node inference process. As we've already discussed, channels are pathways for communicating between nodes within a rule using messages. We must now add additional channels between nodes for unifiable propositions. These channels must ensure compatibility of terms and adjust variable contexts (see Section 4.1). The rule node inference process must

2. We are still considering which kinds of subsumption and instantiation to support and intend that for future work, though it seems that discussed here is the minimal case.

be adapted to check received substitutions for compatibility with each other and combine them when possible (see Section 4.2). Not discussed here are changes to the concurrent processing techniques (Schlegel & Shapiro, 2013a), because none are needed – the techniques are sufficiently general to extend to inferences graphs with quantified terms without modification.

4.1 Channels

Channels are meant to connect all terms which may need to communicate with each other during inference. Within a rule, this includes channels from antecedents to the rule node, and from the rule node to consequents. Since rules now contain quantified terms, we perform unification on all new terms, and add i-channels from rule consequents to unifiable antecedents. This allows rule consequents to communicate their results to appropriate antecedents of other rules for chaining inference. In addition, i-channels are added from propositions to unifiable rule antecedents, and from rule consequents to propositions which might be derived. With these additions, you can think of i-channels carrying messages along the lines of “I have a new instance of me you might be interested in.”

In addition to providing a communication pathway, channels are responsible for ensuring that only messages with substitutions appropriate to the destination pass through, and that they pass through in a form consumable by the destination node (that is, substitutions must be in terms of the proper quantified terms). As in the Active Connection Graph (ACG) (McKay & Shapiro, 1981) – one of the main influences of inference graphs – we augment channels with filters and switches. Filters serve to stop messages with irrelevant substitutions from flowing through the channel, and switches change the substitution’s context to that of the destination term. Both of these are based on bindings which are discovered during the unification process.

When unification is performed, instead of producing an mgu, a factorization is produced which contains *originator bindings*, and *destination bindings*. We call this the match process. The result of a match is a triple, $< D, o, d >$ where D (for *Destination*) is the unifiable formula, and o and d are the originator and destination bindings, respectively. While the exact method of calculation is available in (McKay & Shapiro, 1981), the general idea is that the unification operation is performed as usual, except instead of forming a single substitution, form two – o and d – such that all and only quantified terms in O (the *Originator*, or expression being unified) are given bindings in o, and all and only quantified terms in D are given bindings in d.

The structural subsumption operation for quantified terms co-occurs with unification as part of the match process. A quantified term v subsumes another quantified term u if v’s set of restrictions is a proper subset of u’s. The quantified term u will only match v if the restrictions are the same, or v subsumes u. In addition, the operation to test instantiation of a quantified term, v by another term, t, co-occurs with unification – if t is in every relation in v’s restriction set, it instantiates v. Channels are built from subsumed terms or instances, to the more general term.

Once unification is complete between two terms, the filter and switch can be created. The filter ensures that the incoming substitution is relevant to D by ensuring that for every substitution pair $ft/y \in d$ there is a substitution pair $st/y \in s$ such that either $ft = st$ or st is a specialization of ft, determinable through one-way pattern matching. If a message does not pass the filter, it is discarded. The switch applies the o substitution to the term of each pair in s. This adjusts the substitution to use quantified terms required by the destination. The updated substitution is stored in the passing message.

4.2 Rule Node Inference

As discussed in Section 3.2, when a message reaches a rule node it is converted into a RUI. The RUI is extended to also contain a substitution, which is transferred directly from the incoming message during the RUI creation. Combining RUIs is now more difficult than it was in Section 3.2 because with quantified terms, multiple messages may arrive from each antecedent, each with different sets of bindings for quantified terms, and those bindings may be incompatible. We must therefore ensure that RUIs are compatible before we attempt to combine them. Two RUIs are compatible if their substitutions are. We say that two substitutions, $\sigma = \{t_{\sigma 1}/v_{\sigma 1} \ldots t_{\sigma n}/v_{\sigma n}\}$ and $\tau = \{t_{\tau 1}/v_{\tau 1} \ldots t_{\tau m}/v_{\tau m}\}$, are compatible if whenever $v_{\sigma i} = v_{\tau j}$ then either $t_{\sigma i} = t_{\tau j}$ or one of $v_{\sigma i}$ and t_{τ_j} subsumes or instantiates the other.

To combine RUIs, several structures are available. Which structure is used depends on the logical operation a specific rule node performs. The approaches include a tree-based approach called a P-Tree (Choi & Shapiro, 1992), a hash-map based technique called an S-Index, and a default, which is less efficient.

Rule nodes for logical connectives which are conjunctive in nature can use a structure called Pattern Trees (or, P-Trees) to combine RUIs. A P-Tree is a binary tree generated from the antecedents of the rule. The leaves of the tree are each individual conjunct, and the parent of any two nodes is the conjunction of its children. The root of the tree is the entire conjunction. When a RUI is added to the P-Tree, it enters at the appropriate leaf. The P-Tree algorithm then attempts to combine that RUI with each one in its sibling node. A successful combination (using the compatibility check) is promoted to the parent node, and the process recurs until no more combining can be done because not enough information is present, or the root node is reached. This P-Tree concept is closely related to that of the beta network of a RETE net (Forgy, 1982), which is a binary tree for examining compatibility of tokens used in production systems.

Non-conjunctive rule instances which use the same variables in each antecedent can use a RUI structure called a Substitution Index, or S-Index.[3] This index is a hash-map which uses as a key the set of quantified term bindings, and maps to the appropriate RUI. Given that the same quantified terms are used in each antecedent, each compatible instance will map to the same RUI, which can then be updated and stored again, until enough positive and negative instances have been found.

This unfortunately leaves a large class of rule instances (those with non-conjunctive connectives where the set of quantified terms used in each argument differs) stuck using the default RUI combination, which is inefficient. The default approach to combining RUIs is to compare an incoming RUI with every existing RUI and attempt to combine them. This can result in a combinatorial explosion in the number of RUIs, and should be avoided whenever possible. We consider it important future work to solve this deficiency.

3. Note S-Indexes do not support the compatibility check above, only equality – correcting this is a topic for future work.

4.3 Example

To attempt to illustrate some of the above concepts, consider a KB meant to contain the assertions shown in Figure 4, expressed three different ways: in their English form, in the logical form of $\mathcal{L}_A$ used by CSNePS (on the left side of the figure), and, on the right side, as the individual `wfts` used in the inference graph (Figure 5a). For example, in the graph, `wft5` represents the conditional in item 1 of Figure 4, while `wft1` and `wft4` represent the first and second antecedents of that same item, with the consequent being `wft6`. The two arbitrary individuals `arb1` and `arb2`[4] are the first and second people referred to in the English description of the KB, respectively. Item 3 is represented by `wft7` which uses a new arbitrary individual, `arb3` to represent the arbitrary Person with a listed phone number.

There are three i-channels[5] in Figure 5a between `wft` nodes which are not part of the same complex term (from `wft7` to `wft1`, from `wft11` to `wft1`, and from `wft12` to `wft4`). Under each of the dotted lines drawn on the graph for these channels are three items: v, f, and s, standing for the status of the valve, the filter substitution, and the switch substitution. All of the valves are currently closed, so each v entry has a "c" after it.

In Figure 5b the system wonders whether Stu and Jim communicate (`wft15`). `wft15` is unified with `wft6`, and an i-channel is added from `wft6` to `wft15`, since `wft6` is a consequent, and will be able to report to `wft15` its substitutions. Since the `communicates` relation takes a set of communicators, two filter substitutions are produced: either Stu and Jim can substitute for `arb1` and `arb2`, respectively, or Jim and Stu can substitute for `arb1` and `arb2`, respectively.

Now the system begins trying to infer whether `wft15` should be asserted. It back-chains recursively from `wft15` backward along all u- and i-channels it reaches, opening valves as it goes. Notice in Figure 5c that the valves all say "v = o", since the valves are now open. Substitutions flow from `wft7`, `wft11`, and `wft12` to the antecedents of `wft5` along the red i-channels, and then to `wft5` via the purple i-channels. `wft5` has now received four different substitutions from its two antecedents (`wft12` has two), and must combine them. Since the antecedents of `wft5` are taken in conjunction, a P-Tree is used to perform the combination (see Figure 5d).

The P-Tree, as described in Section 4.2, has a leaf node for each antecedent, and the parent of those (in this case, the root) is the conjunction of them. The incoming substitutions are displayed below the leaf. Each incoming substitution is checked for compatibility with those if its sibling. Since `{Stu/arb2, Dan/arb1}` exists in both leaves, it is promoted to the root, and is a satisfier of the rule. A second pair of substitutions, `{Jim/arb1, arb3/arb2}` and `{Stu/arb2, Jim/arb1}` are a bit harder. Part of the compatibility check the P-Tree performs is to determine if a quantified term and a ground term are structurally similar.[6] In this case, Stu has all the correct arcs to be an instance of `arb3`, so the substitution `{Stu/arb2, Jim/arb1}` is promoted to the root of the tree, and is another satisfier of the rule.

4. `arb1` and `arb2` are different arbitrary Persons because of the special `notSame` restriction. This restriction is enforced outside of the inference graph, and is therefore not represented as a normal restriction in the graph.
5. We have omitted some additional channels which aren't needed for the example to not over-complicate the graph.
6. We are investigating ways to perform this process outside of substitution combination in a more efficient way, including taking cues from Description Logic.

1. If a person calls another person, and the second person answers the call from the first person, those two people can communicate.

```
(if
  (setof
    (calls
      (every x (Isa x Person))
      (every y (Isa y Person) (notSame x y)))
    (answers y x))
  (communicates (setof x y)))
```

```
arb1: (every arb1 (Isa arb1 Person))
arb2: (every arb2 (Isa arb2 Person)
                  (notSame arb1 arb2))
wft2: (Isa arb1 Person)
wft3: (Isa arb2 Person)
wft1: (calls arb1 arb2)
wft4: (answers arb2 arb1)
wft6: (communicates (setof arb1 arb2))
wft5!: (if (setof wft1 wft4) wft6)
```

2. Jim is a Person.

```
(Isa Jim Person)
```

```
wft8!: (Isa Jim Person)
```

3. Jim calls everyone who has a listed number.

```
(calls
  Jim
  (every x (Isa x Person)
           (hasListedNumber x)))
```

```
arb3: (every arb3 (Isa arb3 Person)
                  (hasListedNumber arb3))
wft9: (Isa arb3 Person)
wft10: (hasListedNumber arb3)
wft7!: (calls Jim arb3)
```

4. Dan and Stu are Persons.

```
(Isa (setof Dan Stu) Person)
```

```
wft14!: (Isa (setof Dan Stu) Person)
```

5. Stu has a listed number.

```
(hasListedNumber Stu)
```

```
wft13!: (hasListedNumber Stu)
```

6. Dan calls Stu.

```
(calls Dan Stu)
```

```
wft11!: (calls Dan Stu)
```

7. Stu answers both Dan and Jim.

```
(answers Stu (setof Dan Jim))
```

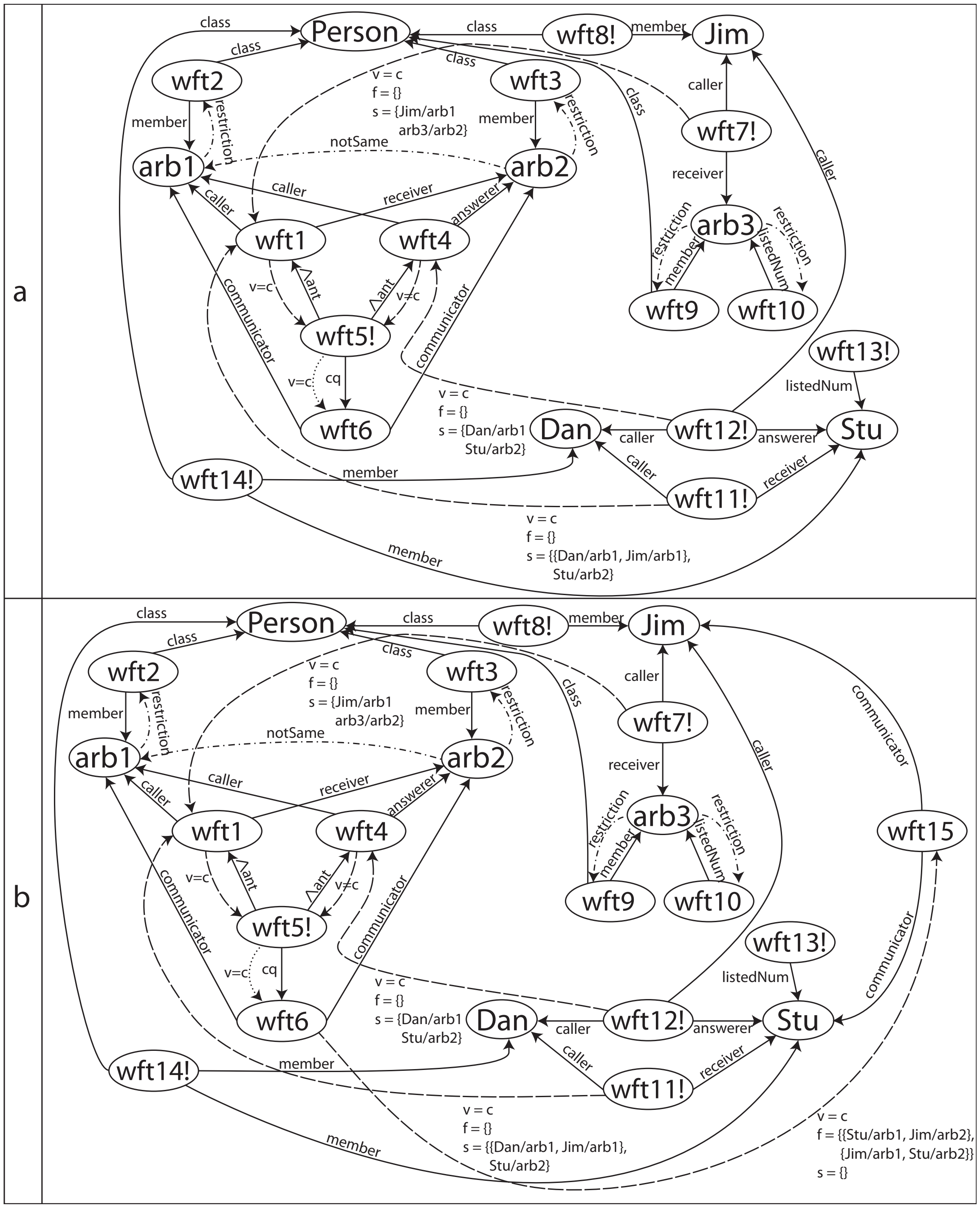
```
wft12!: (answers Stu (setof Dan Jim))
```

Figure 4: An example illustrating concepts from Sections 4.1 and 4.2. The knowledge base is shown in three forms: in English, in the logical form of $\mathcal{L}_A$ (left side of the figure), and, on the right, as the individual `wfts` used in the graph (Figure 5a).

Referring again to Figure 5c, both of the produced substitutions are sent along the blue u-channel to `wft6`. Finally they are both sent along the brown i-channel to `wft15` where only `{Stu/arb2, Jim/arb1}` passes the filter condition, and `wft15` is asserted.

5. "Wh- Question" Answering

When a user is interacting with an agent, or a KR system in general, she often would like to ask a question which has more than a single answer. These questions are generally what we might call in English "wh- questions". In contrast to a question such as "Is Lassie a dog?" which might be answered as we discussed in Section 4.3, we're interested in questions such as "Who are the dogs


a
b
Person
Jim
Dan
Stu
wft1
wft2
wft3
wft4
wft5!
wft6
wft7!
wft8!
wft9
wft10
wft11!
wft12!
wft13!
wft14!
wft15
arb1
arb2
arb3
class
member
restriction
notSame
caller
receiver
answerer
communicator
listedNum
∧ant
v=c
v≠c
cq
v = c
f = {}
s = {Jim/arb1 arb3/arb2}
s = {Dan/arb1 Stu/arb2}
s = {{Dan/arb1, Jim/arb1}, Stu/arb2}
f = {{Stu/arb1, Jim/arb2}, {Jim/arb1, Stu/arb2}}
s = {}

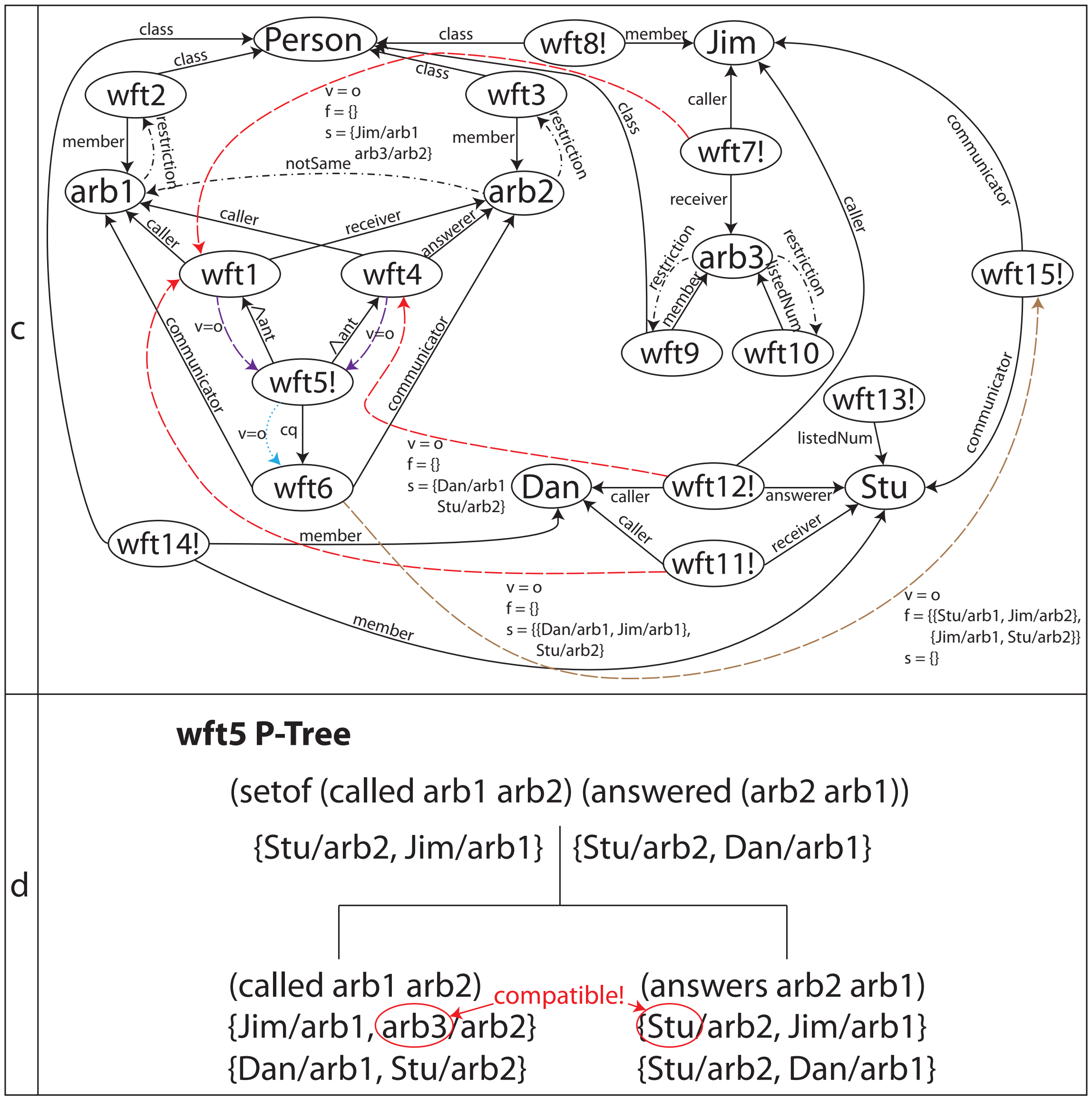


Figure 5: A KB meant to contain the information that: If a person calls another person, and the second person answers the call from the first person, those two people can communicate, Jim calls everyone who has a listed number, Dan calls Stu, Stu answers Dan and Jim, and Stu has a listed number (a). The question is posed whether Stu communicated with Jim (`wft15`, b). Inference is performed (c,d) by retrieving and combining substitutions from related terms, finally inferring that `wft15`, the term representing communication between Stu and Jim, is asserted.

you know about?" In addition, an agent might wish to reason about the questions it has been asked, for example, discussing the questions a certain person asked.

Most all logic-based systems have some method for answering "wh- questions". Perhaps most similar to our own approach is that of ANALOG, which represents questions in the KB, and can answer questions using arbitrary terms – "Since the structure of the question will mirror the structure of the rule [arbitrary term], any rule that is subsumed by a question is an answer to that question" (Ali, 1994). ANALOG is less expressive than $\mathcal{L}_A$, and does not clearly differentiate questions from propositions semantically, which we have found necessary to later reason about questions asked of the system.

In the logical syntax we use to interact with the system, we use question-mark prefixed variable names (such as `?x`), along with a (possibly empty) set of restrictions, to designate a argument which the user is querying over. For example, a user might query the system with `(Isa ?x Dog)` to retrieve all the dogs in the KB. When a question is asked of the system, the question is added to the knowledge base. In order to add a question to the knowledge base, we have created a new kind of quantified term which is used only in "wh- questions". The new quantified term type is called a *question variable* (henceforth *qvar*).

While the query expression or graph may on the surface look like a Proposition, it is clearly not, as it cannot be believed. Groenendijk and Stokhof call the semantic content of interrogative expressions "Questions" (Groenendijk & Stokhof, 2008). Since our question variables are used only in "wh- questions", we call the semantic type for terms containing them WhQuestion. The similarity between the WhQuestion and the Proposition is not lost on us though, both are direct descendants of the semantic type Propositional (as seen in Section 3.1).

As with inference not involving qvars, the result of a query may be an arbitrary or indefinite term. For example, consider a KB which contains:

`(Isa Glacier Cat)`, and
`(Isa (every x (Isa x Tabby)) Cat)`,

meant to mean that Glacier is a Cat, and every Tabby is a Cat. If a user then asked `(Isa ?x Cat)`, the system would respond both that Glacier is a Cat, and that the arbitrary entity who is a member of the class Tabby is a Cat.

6. Focused Reasoning

Humans often consider problems they may not yet have answers for, and push the problem to the "back of their mind." In this state, a human is still looking for a solution to the problem, but is doing so somewhat passively – allowing the environment and new information to influence the problem solving process, and hopefully eventually reaching some conclusion. That is, the examination of the problem persists beyond the time when it is actively being worked on.[7]

7. Understanding this type of problem solving in humans is still active research, what we have discussed is only an intuitive explanation.

This cognitive process is imitated to a limited extent in the ACG which can "activate" a path of nodes. Later assertions meant to use this path must be asserted with forward inference, and that forward inference process will use activated paths exclusively whenever they are available (Shapiro, Martins, & McKay, 1982). The ACG was unable to later deactivate the path of nodes, so the conflation of the specialized forward inference using activated paths with full-forward inference resulted in the need to occasionally throw the graph away as it could interfere with future inference tasks. In addition, activated paths were not extended backward when new rules were added whose consequent unified with antecedents in an activated path. More recent work to achieve a similar result exists in the realm of probabilistic graph reasoning, where methods for a form of focused reasoning where, for example, computation is focused on areas of a model likely to be most important to a query (Chechetka & Guestrin, 2010) are currently being developed.

Inference graphs provide an elegant technique for performing focused reasoning, not completely unlike the intuitive explanation for what happens in humans described above. When a question is asked which the inference graph cannot answer, it recognizes this, leaving the valves in the channels trying to produce the answer open, and taking note of the need to open valves in any new channels which terminate at nodes whose truth value must be known for inference to proceed. Other inference tasks can proceed normally, and new knowledge can be added to the knowledge base. When a new rule, r, is added to the knowledge base, if r has an i-channel from its consequent to a term's antecedent which has its channels open to solve the focused reasoning problem, then r's channels are opened, and backward inference recursively opens channels attempting to derive r's antecedents. When new knowledge is added to the KB which has an i-channel to the antecedent of a rule used in focused reasoning, messages are allowed to flow forward through the graph, to attempt to derive the answer. When a channel is added to the graph which terminates at a node whose truth value is needed to continue inference, its valve is opened, and backward inference is performed recursively from that channel backward.

Consider an agent which has observed an animal named Dumbo and learned that Dumbo has the properties of being Alive, Grey, and Large. It also knows that Dumbo has a Trunk. The agent then wonders if Dumbo is an Elephant (see Figure 6a). The agent currently has no way of judging whether or not Dumbo is an Elephant because it doesn't know what makes an Animal an Elephant. Later on, the agent learns that an Animal is an Elephant if it has a Trunk, and is Alive, Grey, and Large (see Figure 6b). Since the agent was still wondering about whether Dumbo was an Elephant in the "back of its mind", when the new i-channel is added from `wft8` to `wft4`, backward inference is initiated, and the appropriate valves are opened along the channels created by the new rule. Messages then begin flowing forward through the network (see Figure 6c). Instances of antecedents flow along the red i-channels to the newly asserted rule, then along purple i-channels to the rule node itself. Since both antecedents are true, the rule "fires" and an instance is sent along the blue u-channel to the consequent and finally `wft4` is asserted via a message along the brown i-channel. The agent has recognized using focused reasoning that Dumbo is in fact an Elephant.

Where a human probably has some limit to the number of these types of tasks they can perform, we impose no such limits. An interesting future task may be to use this alongside an agent who has a finite number of tasks they can work on, and is "forgetful."

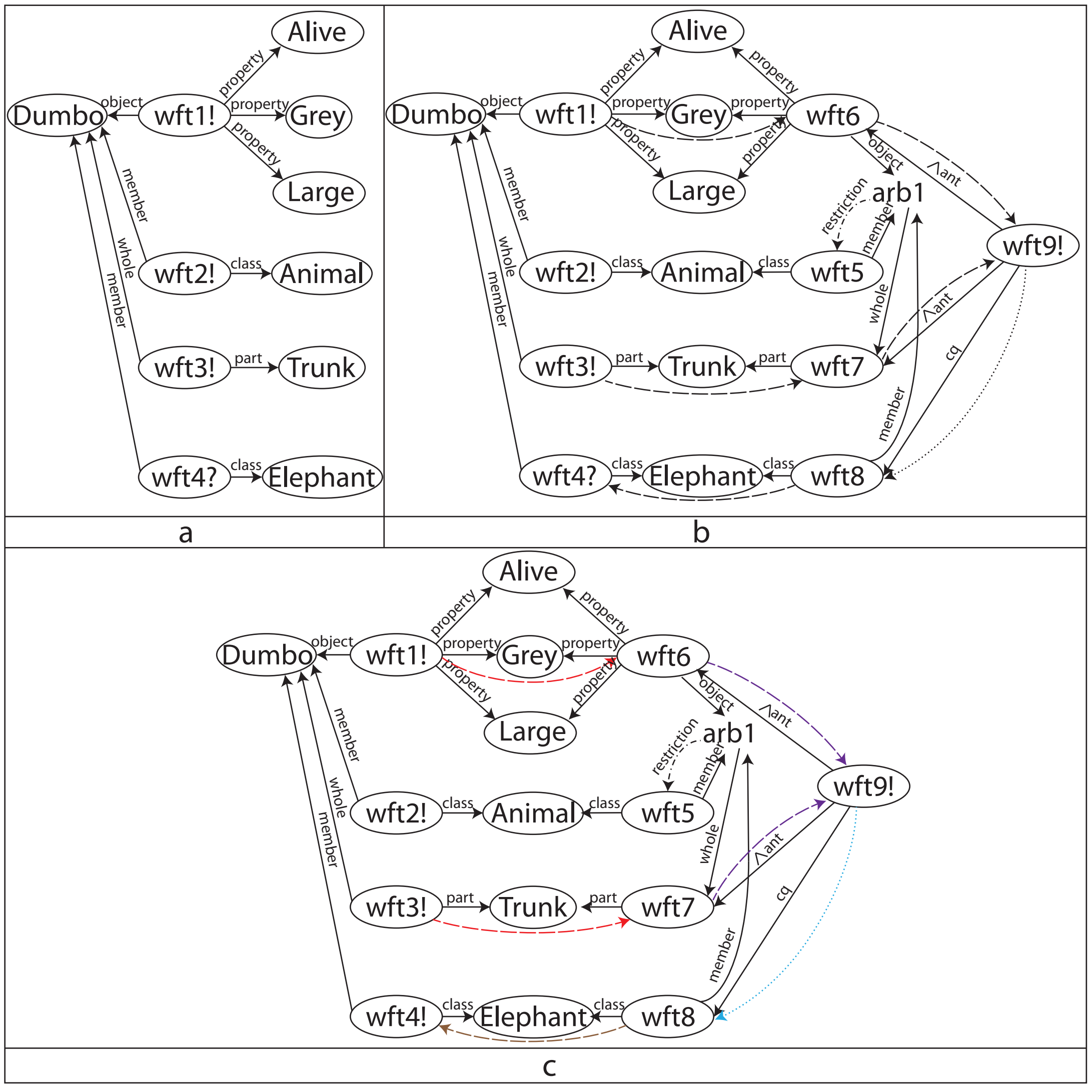


Figure 6: A KB containing assertions that Dumbo has the properties of being Alive, Grey, and Large (`wft1`); has as part of him a Trunk (`wft3`); and is a member of the class Animal (`wft2`). The question (which cannot be answered yet) is asked if Dumbo is an Elephant (`wft4`, a). A question mark, "?", is appended to `wft4` in the graph to show that it is the question being wondered about, though it is a term just like any other in the knowledge base. A rule is added which says that if an Animal (`wft5`) has the properties of being Alive, Grey, and Large (`wft6`), and has a Trunk (`wft7`), then that Animal is an Elephant (`wft8`, b). Since the question about whether Dumbo is an Elephant has already been asked, the valves in the channels for the newly added rule are opened, and messages flow forward through the graph (c), first along red i-channels to the rule's antecedents (`wft1` and `wft7`), then along purple i-channels to the rule itself (`wft8`). The rule is satisfied so the consequent (`wft8`) is notified via the blue u-channel, and finally sends a message to the term which says Dumbo is an Elephant (`wft4`) via the brown i-channel informing it that it is now asserted.

7. Conclusions

The reasoning component of a cognitive system can be implemented as a logical inference system. Inference graphs are one such reasoning component, which with the extensions described here are useful modeling cognitive processes. $\mathcal{L}_A$, an expressive first order logic, allows for reasoning about generic concepts which humans naturally reason about, using arbitrary and indefinite terms. These generic concepts can be used in answering "wh- questions", allowing an agent to tell a user what it knows about not only concrete concepts, but also generic ones. As happens in human conversation often, questions are not always answerable at the time they are posed, due to incomplete knowledge. Inference graphs can consider new knowledge in a way that allows them to answer previously asked questions which may have been unanswerable when they were asked. This might be used to satiate a users, or an agent's own curiosity, as we saw in Figure 6. Together, these can be used to model many of the cognitive functions humans carry out every day, and therefore form the basis of an extremely capable cognitive system.

Acknowledgements

This work has been supported by a Multidisciplinary University Research Initiative (MURI) grant (Number W911NF-09-1-0392) for "Unified Research on Network-based Hard/Soft Information Fusion", issued by the US Army Research Office (ARO) under the program management of Dr. John Lavery.

References

Ali, S. S. (1994). *A "natural logic" for natural language processing and knowledge representation*. Doctoral dissertation, Technical Report 94-01, Department of Computer Science, State University of New York at Buffalo, Buffalo, NY.

Ali, S. S., & Shapiro, S. C. (1993). Natural language processing using a propositional semantic network with structured variables. *Minds and Machines*, *3*, 421–451.

Bona, J. P. (2013). *MGLAIR: A multimodal cognitive agent architecture*. Doctoral dissertation, State University of New York at Buffalo, Department of Computer Science, Buffalo, NY, USA.

Brachman, R. J., & Levesque, H. J. (1987). Expressiveness and tractability in knowledge representation and reasoning. *Computational Intelligence*, *3*, 78–93.

Chechetka, A., & Guestrin, C. (2010). Focused belief propagation for query-specific inference. *Artificial Intelligence and Statistics (AISTATS)*.

Choi, J., & Shapiro, S. C. (1992). Efficient implementation of non-standard connectives and quantifiers in deductive reasoning systems. In *Proceedings of the twenty-fifth hawaii international conference on system sciences*, 381–390. Los Alamitos, CA: IEEE Computer Society Press.

Fine, K. (1983). A defence of arbitrary objects. *Proceedings of the Aristotelian Society* (pp. 55–77).

Fine, K. (1985a). Natural deduction and arbitrary objects. *Journal of Philosophical Logic*.

Fine, K. (1985b). *Reasoning with arbitrary objects*. New York: Blackwell.

Forgy, C. (1982). Rete: A fast algorithm for the many pattern/many object pattern match problem. *Artificial Intelligence*, *19*, 17–37.

Groenendijk, J., & Stokhof, M. (2008). Type-shifting rules and the semantics of interrogatives. In P. Porter & B. Partee (Eds.), *Formal semantics: The essential readings*. Oxford, UK: Blackwell Publishers Ltd.

Iwańska, Ł. M., & Shapiro, S. C. (Eds.). (2000). *Natural language processing and knowledge representation: Language for knowledge and knowledge for language*. Menlo Park, CA: AAAI Press/The MIT Press.

McKay, D. P., & Shapiro, S. C. (1981). Using active connection graphs for reasoning with recursive rules. *Proceedings of the Seventh International Joint Conference on Artificial Intelligence* (pp. 368–374). Los Altos, CA: Morgan Kaufmann.

Schlegel, D. R., & Shapiro, S. C. (2012). Visually interacting with a knowledge base using frames, logic, and propositional graphs. In M. Croitoru, S. Rudolph, N. Wilson, J. Howse, & O. Corby (Eds.), *Graph structures for knowledge representation and reasoning, lecture notes in artificial intelligence 7205*, 188–207. Berlin: Springer-Verlag.

Schlegel, D. R., & Shapiro, S. C. (2013a). Concurrent reasoning with inference graphs. In M. Croitoru, et al. (Ed.), *Lecture notes in artificial intelligence*. Berlin: Springer-Verlag. (In Press).

Schlegel, D. R., & Shapiro, S. C. (2013b). Concurrent reasoning with inference graphs (student abstract). *Proceedings of the Twenty-Seventh AAAI Conference (AAAI-13)* (pp. 1637–1638).

Shapiro, S. C. (2000). An introduction to SNePS 3. In B. Ganter & G. W. Mineau (Eds.), *Conceptual structures: Logical, linguistic, and computational issues. lecture notes in artificial intelligence 1867*, 510–524. Berlin: Springer-Verlag.

Shapiro, S. C. (2004). A logic of arbitrary and indefinite objects. *Principles of Knowledge Representation and Reasoning: Proceedings of the Ninth International Conference (KR2004)* (pp. 565–575). Menlo Park, CA: AAAI Press.

Shapiro, S. C., & Bona, J. P. (2010). The GLAIR cognitive architecture. *International Journal of Machine Consciousness*, *2*, 307–332.

Shapiro, S. C., Martins, J. P., & McKay, D. P. (1982). Bi-directional inference. *Proceedings of the Fourth Annual Conference of the Cognitive Science Society* (pp. 90–93). Ann Arbor, MI: the Program in Cognitive Science of The University of Chicago and The University of Michigan.

Shapiro, S. C., & Rapaport, W. J. (1992). The SNePS family. *Computers & Mathematics with Applications*, *23*, 243–275.

Shapiro, S. C., & The SNePS Implementation Group (2010). *SNePS 2.7.1 user's manual*. Department of Computer Science and Engineering, University at Buffalo, The State University of New York, Buffalo, NY. Available as `http://www.cse.buffalo.edu/sneps/Manuals/manual271.pdf`.

Wang, P. (2006). *Rigid flexibility: The logic of intelligence*. Springer, Dordrecht.

Woods, W. A. (1991). Understanding subsumption and taxonomy: A framework for progress. In J. F. Sowa (Ed.), *Principles of semantic networks*, 45–94. San Mateo, CA: Morgan Kauffman.