# Crossmodal Content Binding in Information-Processing Architectures

Henrik Jacobsson
Language Technology Lab,
DFKI GmbH, Germany
henrikj@dfki.de

Nick Hawes
School of Computer Science,
University of Birmingham, UK
n.a.hawes@cs.bham.ac.uk

Geert-Jan Kruijff
Language Technology Lab,
DFKI GmbH, Germany
gj@dfki.de

Jeremy Wyatt
School of Computer Science,
University of Birmingham, UK
j.l.wyatt@cs.bham.ac.uk

## ABSTRACT

Operating in a physical context, an intelligent robot faces two fundamental problems. First, it needs to combine information from its different sensors to form a representation of the environment that is more complete than any representation a single sensor could provide. Second, it needs to combine high-level representations (such as those for planning and dialogue) with sensory information, to ensure that the interpretations of these symbolic representations are grounded in the situated context. Previous approaches to this problem have used techniques such as (low-level) information fusion, ontological reasoning, and (high-level) concept learning. This paper presents a framework in which these, and related approaches, can be used to form a shared representation of the current state of the robot in relation to its environment and other agents. Preliminary results from an implemented system are presented to illustrate how the framework supports behaviours commonly required of an intelligent robot.

## Categories and Subject Descriptors

I.2.9 [**Artificial Intelligence**]: Robotics

## General Terms

Algorithms, Design

## 1. INTRODUCTION

An information-processing architecture for an intelligent robot is typically composed of a large number of cooperating subsystems, such as natural language analysis and production, vision, motoric skills, and various deliberative processes such as action planning. The challenge addressed in this paper is the production and maintenance of a model of the world for a robot situated in "everyday" scenarios involving human interaction. This requires a method for *binding* together representations provided by a robot's subsystems.

In this paper we will focus on a scenario involving a robot which is able to interact with a human and a set of objects on a tabletop. For example, when faced with a scene containing a red mug, a blue mug and a blue bowl, the robot may be asked to "put the blue things to the left of the red thing". For a system to be able to perform such a task, it must be able to build a representation that connects the (low-level and modality specific) information about the world with the (high-level and amodal) representations that can be used to interpret the utterance, determine the desired world state, and plan behaviour. As actions derived from these processes must be executed in the world, the representations must allow the robot to ultimately access the low-level (i.e. metric) information from which its higher-level representations are derived.

Any design for a system to tackle the above task must address the creation of such representations, and the processes by which they are grounded in the robot's environment. In addition to this, the engineering effort of integrating the various subsystems of the robot with the representations must be considered. After all, since the robot is an engineered system, every component must be put there by means of human effort.

In general, grounding be seen as the process of establishing a relation between representations in two different domains. A special case is when one of the domains is the external world, i.e. "reality":

> The term grounding [denotes] the processes by which an agent relates beliefs to external physical objects. Agents use grounding processes to construct models of, predict, and react to, their external environment. Language grounding refers to processes specialised for relating words and speech acts to a language user's environment via grounded beliefs. [12] (p. 8)

In this paper we will not assume that any component of our system necessarily deals with reality. However, we will


*HRI'08,* March 12–15, 2008, Amsterdam, The Netherlands.

assume that our robot's sensory subsystems may represent things that more or less coincide with aspects of reality relevant to its current task. We also do not consider grounding of only linguistic symbols, nor to physical objects alone, and we do not assume that all representations must relate to physical entities.

In the remainder of this paper we present a design for a subsystem of an information-processing architecture that is able to bind together representations from other subsystems into a single representation shared by the entire system. This binding system, henceforth the *binder*, tackles the problem of creating high-level shared representations that relate back to low-level subsystem-specific representations, as well as addressing the engineering issue of non-intrusively integrating such representations in an implemented system.

The following section will discuss related approaches to similar problems, and identify requirements imposed by the task of creating a situated representation of the world. Following this, the information-processing architecture upon which the binder operates is presented, then the binder is then described in detail. This description is followed by examples of the binder's application in our implemented robot system, and a discussion about its properties in relation to the our stated requirements.

## 2. BACKGROUND AND MOTIVATION

Rather than attempt to address the complete spectrum of problems related to building and grounding representations, we will restrict the problem by addressing the requirements for a *situated* representation of the *current and future state* able to support *deliberative processes* for a specific set of *scenarios*. These restrictions allow us to focus on particular properties of representations that are appropriate for our task domains, i.e. human-robot interaction linked to object manipulation [5] and human augmented mapping of an office environment [19]. In these domains, we are interested in binding together content from separate information-processing subsystems to provide symbols that can be used for deliberation and then action. By deliberation we mean processes that explicitly represent and reason about hypothetical world states, such as a possible interpretation of an utterance or a possible course of action (cf. the deliberative layer of the CogAff schema [15]).

We can specify a number of requirements for deliberative and situated symbols. First, these symbols must be stable for the duration of the deliberative processes they are involved in. For example, a representation of an object from vision should remain stable across multiple image frames if that object is to be involved in a planning process. Second, these symbols must be represented at a level of abstraction appropriate for the processing they will be involved in. For example, objects on a tabletop could be represented in a metric coordinate frame or as abstracted symbols. Each of these representations would be appropriate for reasoning at a particular level of detail. The requirements of stability and appropriate level of detail are closely linked; the level of detail of a particular representation influences its temporal stability. These requirements have been directly informed by the requirements for representations for planning and acting in dynamic, uncertain worlds [18].

Furthermore, since these symbols must be produced by binding content across different concurrently active information-processing subsystems, it is unlikely that the binding of content can happen in a synchronous manner. Perceptual subsystems are typically event driven, and to keep a representation of the state as current as possible, it is important that perceptual information is processed as soon as it is generated. This is importent, for example, in the account of incremental parsing of natural language given in [2]. In this approach the search for possible parses of an utterance is pruned using the context of the current scene. Therefore it is important that any representation of the current state can be incrementally and asynchronously extended as soon as new information is available (i.e. anytime).

Previous robotic systems which are able to bind information from one subsystem to another typically limit this kind of binding to linking linguistic references to objects created from vision. The first system that might even have conceivably encountered the problem was the Shakey system [9]. This translated a constrained set of English language sentences into first order predicate calculus goal statements. Reference here was either non-specific (i.e. "move any box"), or non-ambiguous (each referent that needed to be specifically identified was given a unique name, e.g. "door D"). In making bindings of referents in the goal statement to the objects in the world the non-specific referents allowed lazy binding, so that binding was executed using a unification mechanism at plan execution time. This late binding was only made feasible by the assumption of perceptual reliability, and by the other restrictions given above. However, later systems mostly follow Shakey in their choice of a parsimonious internal language that is a direct mapping onto the qualities of objects that we express relatively straightforwardly in language, and which are naturally stable.

Current approaches, while following this choice of features on which to bind, attempt to bind referents from vision with language using a mixture of deterministic and probabilistic representations, and employing varying levels of abstraction. For example. Mavridis and Roy [8] have a single amodal world model, but one which contains linked deterministic continuous, stochastic qualitative, and stochastic continuous representations. They refer to these as being part of what they call a grounded situation model. In this case the linking is thus essentially not between pairs of properties in vision and language, but between all pairs properties of the same type (colour, position) using a probability distribution over the bindings between. It is, at the time of writing, not yet fully implemented in a robot, and as specified makes no attempt to deal with asynchronous changes to representations in different parts of the system. In other systems [13, 2] binding can occur at a very early stage in processing, allowing even information from the speech signal to influence visual hypotheses for object references, and vice-versa.

Engel and Pfleger [3] approach the problem by gathering all necessary data first, then generating a binding with the highest possible quality. For perceptual priming this approach may not be very fruitful. However, we would argue that it would be a grave mistake to discard earlier work on symbol grounding. If, for example, the systems of [12, 16, 3, 7, 4], could be utilised in one and the same system, we would truly be able to take a step forward as a community. Therefore *non-intrusiveness* is an important requirement on our binding system [10]. In other words, it is important to make it straightforward to integrate existing systems into our binding approach. This requirement also holds for integrating our approach with existing robotic subsystems. This

is in part a requirement on the interfaces to a binding system, they must be kept simple and generic.

To summarise, the main requirements we have on our binder are:

- The symbols produced should be stable,
- they should have the appropriate level of abstraction (i.e. amodal *and* modal),
- they must be generated in an asynchronous, incremental, anytime manner,
- their production must be non-intrusive with respect to existing systems.

# 3. THE ARCHITECTURE

To demonstrate our approach to binding in practise, we have built a robotic system to perform tasks in our tabletop HRI domain. The system has been presented in previous work (e.g. [5]), so we will only give a brief overview here. The design of the system is based on the CoSy Architecture Schema (CAS), a set of rules for designing architecture instantiations in a principled manner. The schema allows a collection of interconnected *subarchitectures* (SAs), each containing a collection of processing components that can be connected to sensors and effectors. Each subarchitecture also contains a *working memory* (WM), which the components use to share information. Only components within a subarchitecture can write to the subarchitecture working memory, but all components can read from any working memory. We also allow for privileged components that can write to any working memory (thus supporting cross-architecture control mechanisms). The schema is enforced in our code using the CoSy Architecture Schema Toolkit (CAST), an open-source, multi-language implementation of CAS [6].

In our implementation we have subarchitectures for vision, communication, manipulation, planning, spatial reasoning, coordination and binding. Together they create a system that can learn and describe object properties in dialogue with a tutor, and also carry out manipulation commands that feature object descriptions based on the learnt visual properties. Each subarchitecture working memory contains specialised representations of the information processed by the attached components. For example, the visual working memory contains regions of interest generated by a segmentor and proto-objects generated by interpreting these regions; the communication subarchitecture contains logical forms generated from parsing utterances; and the spatial reasoning subarchitecture contains abstractions of physical objects and qualitative spatial relationships between them.

# 4. THE BINDER

## 4.1 Overview

The CAS-based architecture provides an ideal test case for the development of a situated representation. Each subarchitecture working memory contains specialised representations, and a subset of these could in principle contribute towards a general representation of the current state. In brief, our approach to tackling this problem has two parts: mapping from specific to general representations, and the fusion of general representations. To enable specialised representations to contribute to the representation of the current state, each subarchitecture must provide a process that translates its specialised representations into more general ones (a process of abstraction). We refer to this process as the subarchitecture's *binding monitor*. The binding monitor provides the abstracted items of information to a separate binding subarchitecture (i.e. the binder) as *binding proxies*. A proxy is essentially a bundle of subarchitecture-specific (modal) information about, e.g., an object, a relationship, a collection of objects etc. The main constituent of a proxy is a set of attribute-value pairs, called *features* (such as colour, ontological category, connections to other proxies etc.). The binder attempts to *bind* proxies together based on whether their defining features agree on a common description. The structures that result from binding proxies are called *union*s, as they essentially contain the union of the features of the bound proxies. The set of unions represents the current best *system wide* hypothesis of the current state. This is based on the assumption that the underlying proxies and features are also the best hypotheses from the corresponding subarchitectures.

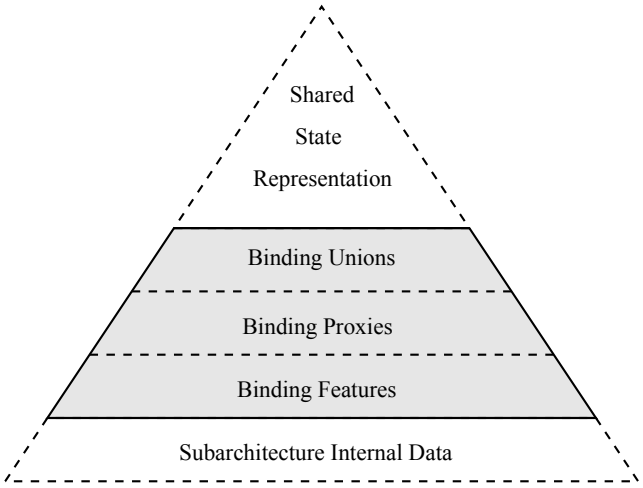


**Figure 1: An illustration of how the binder mediates low-level and modality specific features from the SAs to form a common amodal representation of the world. The SAs are only involved in generation of features and proxies. Everything else is handled by the binder processes. SAs may take into account (or ignore) the common representation formed by the set of unified proxies. Conceptually, the top of the pyramid contains less, more abstract, information than the base. The top level of the pyramid rests solidly on this base, since the features are always referred to from the unified abstracted data.**

The levels of abstraction of the binder and the other subarchitectures are conceptually illustrated in Figure 1. The following sections describe in detail how the above is actually achieved.

## 4.2 Implementation

Our approach to generating a shared representation should not limit what SAs can express (based on the the non-intrusiveness requirement). Therefore the set of possible features is very broad:

DEFINITION 4.1. *A feature space* $\Phi^x \in \mathbf{\Phi}$ *is any data format in the space of all possible data formats,* $\mathbf{\Phi}$.

*$\phi_i^x \in \Phi^x$ denotes an instantiation of a particular representation where $x$ should be interpreted as any feature space name.* □

For example, $\phi_{red}^{ColourLabel} \in \Phi^{Colour}$ denotes the colour "red" in the representation space of colours (the exact implementation of this representation is of no relevance here). In our CAST instantiation, $\mathbf{\Phi}$ corresponds to anything that can be represented by IDL-defined structs (including nested ones).

Information from the SAs is shared as a collection of proxies:

DEFINITION 4.2. *A binding proxy is a structure $p = \langle F_p, u_p \rangle$ where $F_p$ is a set of instantiated features of different types (i.e. $F_p = \{\phi_1^{x_1}, \phi_2^{x_2} \dots \phi_n^{x_n}\}$) and $u_p$ refers to a binding union with which the proxy is bound (see below).* □

The unions should express information from proxies that, by all accounts (cf. Algorithm 1), refer to the same entity. Unions simply inherit the features of the bound proxies and are defined as:

DEFINITION 4.3. *A binding union is a structure $u = \langle F_u, \mathbf{P}_u \rangle$ where $\mathbf{P}_u$ refers to the subset of proxies unified by the union $u$ and $F_u$ is defined as the union of the features in all proxies in $\mathbf{P}_u$.* □

The problem for the binder is to assess whether two proxies are matching or not. By matching we mean that they should refer to the same thing. To do this, all new or updated proxies are compared to all unions on the basis of their respective features. The basis of this comparison is that each pair of feature types has an associated comparator function:

DEFINITION 4.4. *A feature comparator is a function $\Delta : \Phi^x \times \Phi^y \rightarrow \{true, false, indeterminate\}$ returning a value corresponding to whether two feature instances are equivalent (or similar enough) or not. The comparator can also choose to not return a definite answer if the answer is undefined, or the uncertainty is too big (i.e. indeterminate).* □

Obviously, *indeterminate* is the only answer most such comparators can return, e.g. the comparison of a $\Phi^{Colour}$ and a $\Phi^{Position}$ is likely undefined[1]. However, for many pairs of features there exist informative comparators. For example, features such as linguistic concepts can be compared to other concepts (with ontological reasoning [3]) or physical positions can be compared to areas.

DEFINITION 4.5. *Two feature spaces $(\Phi^x, \Phi^y)$ are* comparable *iff $\exists(\phi_i^x, \phi_j^y) \in (\Phi^x, \Phi^y)$ such that $\Delta(\phi_i^x, \phi_j^y) \neq indeterminate$.* □

The more pairs of features from different SAs that are comparable, the more likely it is that proxies from these SAs will be accurately matched.

To compare a proxy and a union, the corresponding feature sets are the basis for scoring:

DEFINITION 4.6. *The binding scorer is a function $S^+ : \mathcal{P} \times \mathcal{U} \rightarrow \mathbb{N}$ where $\mathcal{P}$ and $\mathcal{U}$ denote the set of all proxies and unions respectively and*

$$S^+(p,u) = \sum_{\phi_i^x \in F_p} \sum_{\phi_j^y \in F_u} \begin{cases} 1 & \text{if } \Delta(\phi_i^x, \phi_j^y) = true \wedge \phi_i^x \neq \phi_j^y \\ 0 & \text{otherwise} \end{cases}$$

*where $F_p$ and $F_u$ are the feature sets of $p$ and $u$ respectively.* □

Note that identical features are not counted. This to prevent a union getting a higher score just because it is compared to one of its member proxies (this would sometimes prevent a proxy switching to a better union). The number of feature mismatches is also counted (i.e. with *true* replaced with $false$ in $S^+$). That function is here denoted $S^- : \mathcal{P} \times \mathcal{U} \rightarrow \mathbb{N}$.

It is important to state that $S^+$ and $S^-$ are implemented *asynchronously* with respect to the comparators. Until a comparator has returned an answer, $S^+$ and $S^-$ will simply assume that the answer is neither *true* or $false$, i.e. *indeterminate*.

$S^+$ and $S^-$ are the basis for selecting the best among all unions for each new or updated proxy. This is conducted by the function `bestUnionsforProxy` described in Algorithm 1. The result of $best$ = `bestUnionsforProxy` is a set of zero, one or more unions. If $|best| = 0$ then a new union will be created for the proxy $p$ alone (i.e. with all the features of $p$). If $|best| = 1$, then the proxy is bound to that union.

When $|best| > 2$ we are faced with a *disambiguation* problem. To avoid deadlocks in such cases the binder selects a random union from *best* for binding. However, bindings are *sticky*, meaning that if an already bound proxy subsequently matches a union in a larger "best"-list, then it will not switch to any of those unions. This to avoid excess processing in, and signalling from, the binder. This also helps to satisfy our requirement for symbols to be stable as far as possible. Disambiguation problems cannot be solved by the binder itself, but it can request help from others SAs. This may result, for example, in the communication SA initiating a clarification dialogue with a human tutor (cf. Section 5.4).

```
bestUnionsforProxy(p,U)
Input: A proxy, p, and the set of all unions, U.
Output: Best union(s) with which a proxy should bind.
begin
    best := ∅;
    max := 0;
    for ∀u ∈ U do
        if S⁻(p,u) = 0 ∧ S⁺(p,u) > max then
            best := {u};
            max := S⁺(p,u);
        else if S⁻(p,u) = 0 ∧ S⁺(p,u) = max then
            best := best ∪ {u};
        end
    end
    return best;
end
```

**Algorithm 1: The algorithm which computes the set of best candidate unions for being bound with a new or updated proxy (see definitions 4.1-4.6 for an explanation of the notations).**

## 4.3 Relations and Groups

The proxies and unions described so far have been assumed to roughly correspond directly to physical objects. They may also correspond to more abstract entities as well. To support this, two special proxy types are implemented

[1] Of course, in the implementation, such undefined comparators are never invoked. Mathematically, however, this is exactly what happens.

in a slightly different manner: proxies denoting groups of proxies, and proxies denoting relationships between proxies.

Since proxies contain features that are of any representable type, proxies can also have features attributable to groups and relations, e.g. cardinality and relative metric information respectively, and explicit references to relating proxies. Currently we handle groups in a fairly simple yet direct way: a special kind of "group proxy" is created exactly like an ordinary binding proxy with all the features that the members of the group have in common (e.g. "the blue balls to the left of the mug" creates a group with features $\phi_{ball}^{Concept}$ and $\phi_{blue}^{ColourLabel}$ and with a spatial relation $\phi_{left\_of}^{SpatialRel}$-proxy to the $\phi_{mug}^{Concept}$-proxy. A separate process in the binding SA (the "group manager") then spawns off individual proxies which inherit the features of the group proxy. Every time an individual is bound to something, a new proxy is spawned[2]. To all the other processes, the individuals appear as an endless supply of normal proxies.

Relation proxies are implemented in a similar way as standard proxies, but with additional features indicating the other proxies involved in the relation. Features of relation proxies are thus compared using the same mechanism that compares the features of standard proxies. For example, spatial metric features, e.g. $\phi_{(x,\vec{y},z)}^{\mathbb{R}^3}$, could in principle be compared to a linguistic feature describing the same relation, e.g. $\phi_{left\_of}^{SpatialRel}$. It has turned out that features that link relations to normal proxies and vice versa make the scoring inefficient. Therefore, a separate scoring scheme similar to that in definition 4.6 is used to assess how well proxies match to unions w.r.t. their relational context (but due to space limitations, this is left out of the description).

# 5. EXAMPLES

To illustrate how our binder supports a number of behaviours typically required of robots that interact with humans, the following sections present a number of examples taken from our implemented system. These examples refer to the implemented system and subarchitectures described in Section 3.

## 5.1 Visual & Spatial Reference Resolution

Perhaps the most common use of information fusion systems is to interpret linguistic references in terms of visual information (cf. Section 2). Our binder handles this task as an instance of a more general problem of information fusion. We will here consider the simple situation where we have a red object and two blue objects on the table. The objects are arranged in a straight line of alternating colours. The human then asks the robot to "put the blue objects to the left of the red objects".

We will start our example in the visual subarchitecture, where change detection, tracking and segmentation components create representations of the objects in the scene. These objects have 3D poses and bounding boxes and a number of slots for visual properties such as colour, shape and size. These slots are filled by a recogniser that has been previously trained (see Section 5.3) using input from a human trainer [14]. For this example we assume the recogniser correctly extracts the colours of the objects as red and blue. When the scene becomes stable (determined by the change detector) the visual subarchitecture binding monitor creates a proxy for each of the currently visible objects. As the visual property recogniser processes the objects, the monitor updates the proxies with features reflecting these properties. This is an incremental process, so the visual proxies are updated asynchronously as the objects are processed. At this point only the visual proxies are present in the binding working memory, each one is bound to its own union.

The presence of objects in the visual working memory is also noticed by the components in the spatial subarchitecture. These abstract the objects as points on the tabletop, and the spatial binding monitor creates a proxy for each. These proxies are tagged with the ID of the visual proxy for the corresponding object so they are bound correctly[3]. Concurrently with the proxy creation, qualitative spatial relations between the spatial objects are added to working memory. These are generated by components using potential-field-based models of spatial relations [1]. In our example the two blue objects are to the left and to the right of the red object respectively. They are both also near the red object (but not near each other). As these relations are added, the spatial binding monitor reflects them on the binding working memory as relation proxies between the spatial proxies. The binder uses these as the basis of relations between the unions featuring the spatial proxies. This provides our basic model of the current state.

When the human speaks to the robot, a speech recognition module in the communication subarchitecture is triggered. The resulting speech string is written to the communication working memory. This triggers a cycle of deep sentence analysis and dialogue interpretation, yielding a structured logical description of the utterance's content. From this structure the communication binding monitor generates communication proxies for the discourse referents from the utterance and the relations between them. These proxies include features that can match against both those attached to visual proxies (colour, shape and size), and those attached to spatial proxies (relations based on spatial preposition). In the example two proxies are generated: one normal proxy for the red object, and one group proxy for the blue objects. The binder uses the features of these communication proxies to bind them into unions with the visual and spatial proxies. In the example the $\phi_{red}^{ColourLabel}$-proxy is bound together with the visual and spatial proxies relating the red object, and the $\phi_{blue}^{ColourLabel}$-proxies spawned from the corresponding group proxy (see Section 4.3) are bound with the remaining proxies for the blue objects. This provides the system with an interpretation of the utterance in terms of the visual scene.

In this example, the process of reference resolution involves simply ensuring that the communication proxies referring to visual entities (i.e. those referring to objects in the tabletop scenario) are bound to unions that have a visual component. If the utterance contains spatial language, then relation proxies are generated by the communication binding monitor. This causes the binding process to bind proxies via the relations between proxies as well as the features of single proxies. Failure to bind proxies can trigger a number of different processes, as described in 5.4.

[2]With some obvious limitations to allow finite groups and to prevent excess proxies being generated when members of different groups merge.

[3]A similar, but more general, functionality could be generated by matching location-derived features.

## 5.2 Planning and Execution

Once the system has successfully interpreted an utterance, it must also generate some behaviour. In addition to creating proxies, the interpretation of the utterance also produces information about the purpose of the utterance. In this case the utterance is determined to be an instruction, and this causes the coordination subarchitecture to generate a motive to act on the instruction. This process involves reinterpreting the utterance as a planning goal in MAPL, the multi-agent planning language used by the system. This interpretation is carried out by a general-purpose process that maps between verbs and planning operators [1]. Once the system has a goal, the planning subarchitecture is used to generate and execute a plan to satisfy it.

To create a plan, the planning subarchitecture needs to generate a description of the initial state for the planner to operate on. This is done by translating directly from the binding unions (accessed through the communication proxies stemming from the utterance) and their features into objects and fact descriptions in MAPL. Once a plan has been created, the execution components in the planning subarchitecture start working through the plan triggering execution steps followed by execution monitoring steps.

In our current system we only have actions related to manipulation (pick up and put down), so all plan actions are forwarded to the manipulation subarchitecture. As the planning process used binding unions as input, the plan actions are also expressed in terms of these unions. The manipulation subarchitecture cannot operate on the symbolic representation used by the planner, but requires the detailed metric information generated by processes in the visual subarchitecture to support grasping. By following links from the input unions in the planned action via the visual proxy to the object information in the visual subarchitecture, the manipulation processes get access to the necessary metric data. As the processes in the visual subarchitecture run constantly, this metric information is always kept consistent with the world. The binding structures, however, remain stable across these changes (unless they are significant enough to alter the spatial relations). This example demonstrates how our two-level approach to representing the current state of the world supports access to low-level information via high-level symbols, whilst allowing these symbols to remain stable as the low-level information they are derived from is updated.

In our example, the plan involves a single pair of pick and put actions. The blue object on the right of the scene is moved to the left of the red object. The other object is already to the left of it so it is not moved. After each object is moved the planning subarchitecture triggers an execution monitoring step. This step involves creating a new representation of the current state from the unions on binding working memory, and comparing them to the state predicted by the plan. For the monitoring step in the example to complete successfully, the unions must reflect that the moved blue object is now at a position to the left of the red object. Being able to monitor for such abstractly-specified conditions demonstrates the benefit of generating symbolic states on demand from a dynamically updated representation of the current state.

## 5.3 An Example of Interactive Learning

Our example system learns the visual properties of objects through dialogue with a human tutor (an interaction that can take many forms [17]). The tutor trains the system with sentences such as "this is a large red thing" and "this is to the left of the blue thing", and the visual properties are ultimately learnt by a continuous learning system [14]. Our approach to binding naturally supports the creation of training examples for this learning system. When an object is placed in front of the robot, the visual subarchitecture processes the object as described previously, ultimately creating a visual proxy for it. When the tutor makes an assertion about the object (or relation), we use recency information to bind the communication proxy for the deictic reference to the newest visual proxy[4]. The communication proxy contains binding features for all of the adjectives used in the utterance. When the visual subarchitecture binding monitor is informed its proxy has been bound into a union (via CAST change events), it inspects the union to see what features are present that it didn't add itself. These features represent information about the object from other modalities that the visual subarchitecture can choose to learn. Currently, we take a fixed subset of features present in the union and use them to generate input to our learner. In theory these restrictions could be removed and features provided by other modalities could be used by any subarchitecture to learn cross-modal information. This simple way of driving cross-modal learning systems demonstrates a benefit of both our approach to binding and of working on learning in integrated systems.

## 5.4 Generation of Clarification Events

It is not always possible for the binder to find unique bindings for proxies. For example, consider a scene with two red objects on the table which cause two visual $\phi_{red}^{Colour}$-proxies to be created and bound into separate unions. The human then asks the robot to "pick up the red thing", creating a $\phi_{red}^{ColourLabel}$-proxy from the discourse. In this situation `bestUnionsforProxy` will return a set containing the two visually red unions. In the near future we plan to use situations such as this as a general purpose trigger for generating *clarification behaviour*. For example, consider the case where the visual proxies (and thus their unions) have some mismatching features that separate them (e.g. $\phi_{round}^{Shape}$ and $\phi_{square}^{Shape}$). In this case, this could lead directly to the system generating a goal to determine if the object being referred to by the human has one of the mismatching features. In this example the robot could ask a question about the distinguishing feature (e.g. "do you mean the *round* red thing?"). However, in principle the general purpose nature of the binding system means that any subarchitecture that can provide a particular binding feature could satisfy such a request for information (i.e. not only dialogue).

The situation where the ambiguous unions have matching feature sets raises a different type of clarification problem. Rather than generating a need for a particular type of feature information to clarify a binding, resolving this situation requires a direct reference to the target object to allow binding. For example, the robot may have to ask "which red thing do you mean?", "do you mean this one?" (whilst pointing), or "do you mean the one on the left?". Alternatively the binding system could draw on information from

[4]In this instance we are using recency as a substitute for a more complex process of reference resolution.

other modalities to determine things such as the likelihood of the object being involved in a pick-up action (e.g. a feature $\phi_{true}^{Reachable}$), or the salience of the object given the human's perspective on the scene. Building support for the sharing of such information via proxies allows general notion of salience and attention to be built into the system.

# 6. DISCUSSION

In the following sections we discuss the properties of the representation and binding system presented previously, including how it relates to our original requirements.

## 6.1 Modal and Amodal Representations

As is apparent in the planning example above (Section 5.2), from the point of view of a particular SA, a union is an *amodal* entity. But despite this, it also contains a set of *modal* properties some of which have semantics for particular SAs. The binder mixes amodal and modal representations such that modality-independent proxies and unions can be used for symbolic processing while at the same time they contain references to modal representations, i.e., via the features. Moreover, if a feature space is used which supports the ability to refer to data *inside* local SA WMs, data types that have not been declared as features can still be shared with all other SAs if required.

## 6.2 Lazy Binding and Locally Stable Symbols

In our binder, the only thing an SA typically needs to keep track of, once a binding proxy is created, is the proxy itself. Once created as a candidate for binding, the proxy indeed *acts* as a proxy which mediates information from its union. This means that any SA-internal symbols can be made isomorphic to the indexes of the proxies and the SA does not need to necessarily take into account whether the proxy is in a union or not.

This can be a very powerful simplification for many types of SAs. For example, consider the navigation dialogue scenario where the user tells the robot to "go to the kitchen" [19]. Now, if the robot has yet to discover the kitchen, the discourse referent proxy containing $\phi_{kitchen}^{Concept}$ can not really be bound to any kitchen in the map. As soon as some other process identifies and defines a kitchen in the map, the utterance's $\phi_{kitchen}^{Concept}$-proxy can be bound. Whether or not this binding takes place, the $\phi_{kitchen}^{Concept}$-proxy remains intact and can still be referred to internally in the same way. From the point of view of an individual SA, unions just provide potentially richer descriptions of its own proxies.

## 6.3 Scalability

The theoretical properties of the binder are irrelevant if it cannot be implemented in an effective way. Potentially, the binder may become a bottleneck in an architecture since it may receive features and proxies from all involved subarchitectures[5]. To overcome this, we have implemented the binder as several smaller components, each responsible for basic and well-defined tasks (e.g. invoking and collecting the results of comparators, generating the unions based on feature scores etc.). All of these components can be replicated and put on different physical nodes, sharing the computational load. Moreover, the feature comparisons are made externally to the binder and thus the computational load is further distributed[6].

The problem of making the binder scalable is in part addressed by the role of abstraction in the system. The data "closest" to the binder (i.e. the unions) are abstracted from the much more abundant features (which are primarily processed by the SAs themselves). This means the binder only has to operate on an abstracted subset of all of the information in the system.

## 6.4 Incremental Asynchronous Binding

As mentioned in Section 2 it is desirable to do both early and anytime binding. We achieve this in our implementation by allowing all components to operate on the data asynchronously. This makes binding quite efficient since any processing tasks (feature comparisons, scorings, union creation etc), will be carried out as soon as is possible.

For example, in Section 5.1, the visual and spatial proxies are initially bound as basic object abstractions. Following this, SA-specific components gradually add more information about the objects (visual properties and spatial relations), which cause the proxies, and their unions, to be asynchronously updated with features and relations.

The anytime properties of the binder also mean that any comparison that is finished early may help in forming unions before any additional comparisons are made. Thus unions may be formed at an early stage and then refuted if conflicting information arrives later. This may create an overhead given that incorrect unions are temporarily created. SAs with higher quality demands can always wait until the bindings have "settled".

## 6.5 Demands on Subarchitectures

From an engineering point of view, subarchitecture designers have to provide a number of things to support the binding process: 1. the feature definitions (if not already existing), 2. a binding monitor component that analyses local SA WM content and generates and mediates proxies onto the binding working memory, and 3. the comparators.

The comparators can be based upon any kind of computational process from simple equivalence testing like string matching to ontological, spatial or visual reasoning etc. The comparators may also be learnt models and can even be learnt online, on data extracted from unions, while the binder is operating (as presented in Section 5.3). Moreover, a comparator may be *context sensitive*, i.e. it can take into account all other information on the binding WM to make its assessment (cf. [11]). It is also possible that the comparator itself triggers the SA to generate more features to complete a proxy's description. There are many possibilities since few limitations are imposed by the design of the binder.

The integration with the binder is fairly non-intrusive in the sense that none of the things that need to be provided should have any implications on any other part of the SA. Minimally, the SA only needs to write features and proxies, and does not have to process what is happening on the binder at all.

A slightly deeper integration problem occurs when a SA needs to utilise the contextual information represented by the unions (e.g. for priming in incremental parsing). This is, however, arguably non-intrusive as well, as all features that

[5]This is one reason why SAs should be conservative about generating proxies.

[6]In the implementation some trivial comparisons are actually handled internally in the binder.

are unknown to the SA in question can safely be ignored (cf. section 6.1).

The feature set space is also highly open-ended. An added feature definition will not affect, nor depend on, the earlier features in any way. Every subarchitecture can and will only deal with the features it knows about (i.e. it is non-intrusive w.r.t. data formats). This means that there is a fairly low cost to adding features into the system.

One problem for some designers may be that expressive models of beliefs (e.g. Bayesian) are being robbed of their expressiveness when a comparator can only return three values. This is however a better situation than the opposite. There is also nothing that prevents a comparator from reasoning about degrees of belief up to the point of the final comparison decision.

In order for the binder to perform well, the designers need to be conservative. For example, proxies should not be generated excessively (for example, just to see if they will be bound or not) since it may disturb other SAs. And new feature types should primarily only be introduced if also a comparator for this feature can be defined.

If conservatism is not employed, the binder will not perform well. Since features and comparators are representable in very open-ended formats, the SA designer has very few limitations in what can be done. This is of course an advantage in many cases. But many creative interpretations of "features", "proxies" and "comparators" will simply not yield desired results. For example, conflicting features can be inserted into a proxy, but that violates the proxy-as-best-hypothesis assumption.

Another problem is if the SAs can only provide features that are SA specific and incomparable with most features from other SAs. In such cases the binder would not be able to form any unions. It is thus important to have comparable features in mind when integrating SAs into an architecture.

# 7. CONCLUSION

In this paper we presented a method for generating a stable, yet asynchronously updated, representation of the current state of the world for a situated information-processing system such as an intelligent robot. The amodal representation emerges from the incremental fusion of information abstracted from modal data, and satisfies the requirements we specified for such a system. Although our system has been fully implemented, we have yet to evaluate it experimentally. In place of this we illustrated our system with a number of examples from the scenarios we are tackling.

# 8. ACKNOWLEDGEMENTS

This work was supported by the EU FP6 IST Cognitive Systems Integrated Project "CoSy" FP6-004250-IP.

# 9. REFERENCES

[1] M. Brenner, N. Hawes, J. Kelleher, and J. Wyatt. Mediating between qualitative and quantitative representations for task-orientated human-robot interaction. In *Proc. IJCAI '07*, 2007.

[2] T. Brick and M. Scheutz. Incremental natural language processing for hri. In *Proc HRI '07*, pages 263–270, 2007.

[3] R. Engel and N. Pfleger. Modality fusion. In W. Wahlster, editor, *SmartKom - Foundations of Multimodal Dialogue Systems*, Cognitive Technologies, pages 223–235. Springer, July 2006.

[4] B. Fransen, V. Morariu, E. Martinson, S. Blisard, M. Marge, S. Thomas, A. Schultz, and D. Perzanowski. Using vision, acoustics, and natural language for disambiguation. In *Proc HRI '07*, pages 73–80, New York, NY, USA, 2007. ACM Press.

[5] N. Hawes, A. Sloman, J. Wyatt, M. Zillich, H. Jacobsson, G. Kruijff, M. Brenner, G. Berginc, and D. Skocaj. Towards an integrated robot with multiple cognitive functions. In *Proc. AAAI '07*, 2007.

[6] N. Hawes, M. Zillich, and J. Wyatt. BALT & CAST: Middleware for cognitive robotics. In *Proc. IEEE RO-MAN 2007*, pages 998 – 1003, August 2007.

[7] G.-J. Kruijff, J. Kelleher, and N. Hawes. Information fusion for visual reference resolution in dynamic situated dialogue. In E. Andre, L. Dybkjaer, W. Minker, H. Neumann, and M. Weber, editors, *Proc. PIT '06*, pages 117 – 128, 2006.

[8] N. Mavridis and D. Roy. Grounded situation models for robots: Where words and percepts meet. In *IEEE/RSJ International Conference on Intelligent Robots and Systems 2006*, pages 4690–4697. IEEE/RSJ, October 2006.

[9] N. J. Nilsson. Shakey the robot. Technical Report 323, AI Center, SRI International, 1984.

[10] D. L. Parnas. On the criteria to be used in decomposing systems into modules. *Commun. ACM*, 15(12):1053–1058, 1972.

[11] D. Roy. Grounding words in perception and action: Insights from computational models. *Trends in Cognitive Science*, 9(8):389–96, 2005.

[12] D. Roy. Semiotic schemas: A framework for grounding language in action and perception. *Artificial Intelligence*, 167(1-2):170–205, 2005.

[13] D. Roy and N. Mukherjee. Towards situated speech understanding: visual context priming of language models. *Computer Speech & Language*, 19(2):227–248, 2005.

[14] D. Skočaj, G. Berginc, B. Ridge, A. Štimec, M. Jogan, O. Vanek, A. Leonardis, M. Hutter, and N. Hawes. A system for continuous learning of visual concepts. In *International Conference on Computer Vision Systems ICVS 2007*, Bielefeld, Germany, 2007.

[15] A. Sloman. Varieties of affect and the CogAff architecture schema. In *Proc. the AISB'01 Symposium on Emotion, Cognition and Affective Computing*, pages 1–10, 2001.

[16] L. Steels. Semiotic dynamics for embodied agents. *IEEE Intelligent Systems*, 21(3):32–38, 2006.

[17] A. L. Thomaz. *Socially Guided Machine Learning*. PhD thesis, Massachusetts Institute of Technology, May 2006.

[18] S. Wood. *Planning and Decision Making in Dynamic Domains*. Ellis Horwood, 1993.

[19] H. Zender, P. Jensfelt, Óscar Martínez Mozos, G.-J. M. Kruijff, and W. Burgard. An integrated robotic system for spatial understanding and situated interaction in indoor environments. In *Proc. AAAI '07*.