Crossmodal Content Binding in Information-Processing Architectures

Henrik Jacobsson[1], Nick Hawes[2], Geert-Jan Kruijff[1], Jeremy Wyatt[2]

[1]Language Technology Lab, DFKI GmbH, Germany

[2]School of Computer Science, University of Birmingham, UK

Dec 2007, Aveiro, LangRo symposium

Binding

Jacobsson, Hawes, Kruijff, Wyatt

Introduction

The Problem

Motivation

The Binder

Discussion

Outline

The CoSy Project

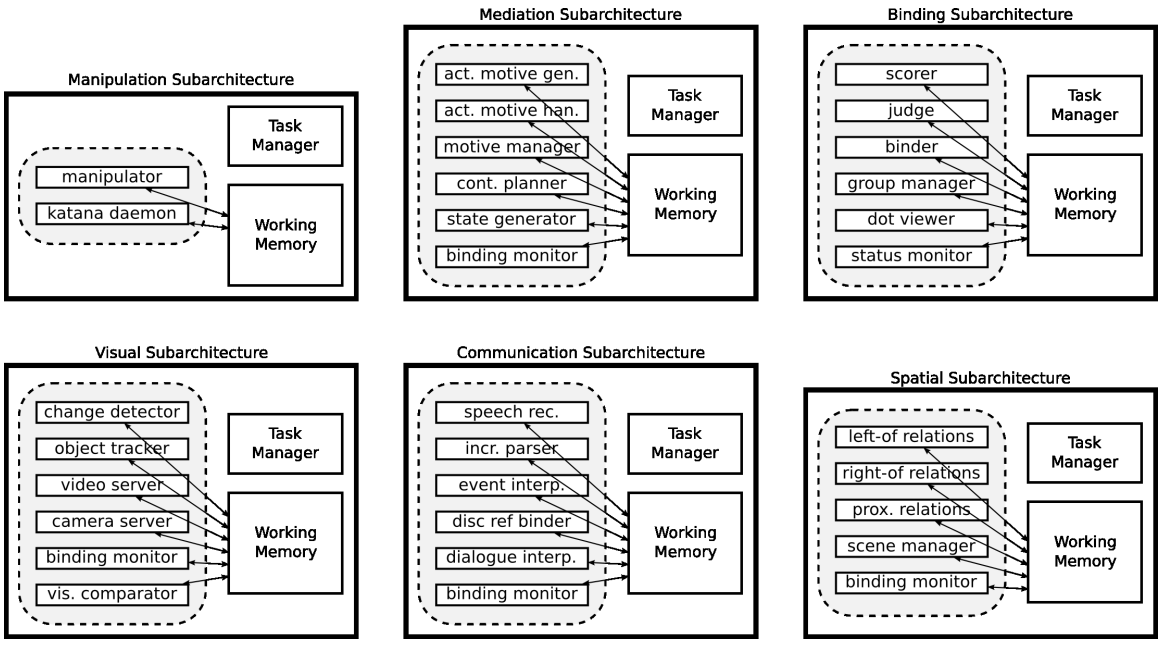
EU FP6 IST Cognitive Systems Integrated project Cognitive Systems for Cognitive Assistants - CoSy

The main goal of the project is to advance the science of cognitive systems through a multi-disciplinary investigation of requirements, design options and trade-offs for human-like, autonomous, integrated, physical (eg., robot) systems, including requirements for architectures, for forms of representation, for perceptual mechanisms, for learning, planning, reasoning and motivation, for action and communication

The CoSy Architecture Schema Toolkit (CAST)

- One of the main focus of the research in CoSy is to investigate the design space of cognitive robotics
- The architecture toolkit aims at making it possible to investigate a range of possible *instances* of architectures
- An architecture consists of several uniformly designed subarchitectures dedicated to vision, planning, communication, mapping etc.
- The main challenge is the integration effort
  - How to communicate between subarchitectures
  - What to communicate between subarchitectures
  - What to do with information from other subarchitectures
  - When to communicate

CAST Example



The Binding Problem

- a.k.a. symbol grounding
  - but... “[The binder does] not explicitly deal with reality”
- That will not eliminate all problems:
  1. Find a common ground for representing information from different sensory modalities and deliberative processes
  2. Find a format that facilitates integration of existing and future implementations of subarchitetures
  3. Consider other binding problems than those related to language
  4. Robustness against ”chaotic” dynamics important

Requirements

1. Appropriate level of abstraction
2. Nonintrusive and simple
3. Stable symbols
4. Asynchronous, anytime, incremental production of bindings

Requirements
1. Level of abstraction

- Dilemma: amodal or modal information? Both make sense! So we support both!
- The entities in our scenarios typically involve
  - Objects (and groups of objects)
  - Actions
  - Relations
- We represent these as entities with sets of describing properties, *binding features*
- These entities are called *proxies* (why? hang on)
- The information fusion of crossmodal contents build upon the assumption that subarchitectures have proxies that may refer to the same entity
  - Independent of temporal or spatial frame

Requirements
2. Nonintrusiveness

- A subarchitecture only needs to provide:
  - Binding feature *definitions*
  - A binding *monitor* component which create appropriate proxies
  - *Comparators*, that compare pairs of features

Requirements
Nonintrusiveness of Binding Features

- A subarchitecture can have very specialized representations, e.g.
  - visual features
  - spatial representations
  - linguistic modifiers
  - etc.
- The depth of description would be restricted without them
- Translation into common description is costly and lossy
- A binding feature can therefore in principle be *anything*
  - I.e. anything you can represent in a Java or C++ class
  - If your subarchitecture come across a feature which it doesn't understand, it can only ignore it

Requirements
The Relative Intrusiveness of Binding Monitors

- The monitors should react to internal data, and make a proper presentations of it in the form of proxies
  - Intramodular binding (e.g. discourse referents or spatial reasoning)
  - Present the currently best hypothesis about objects, actions and relations (i.e. possibly incrementally)
  - Monitors can be context aware (e.g. to withold irrelevant data)
- Monitors should present data that is likely to be relevant to the task... (not so easy)

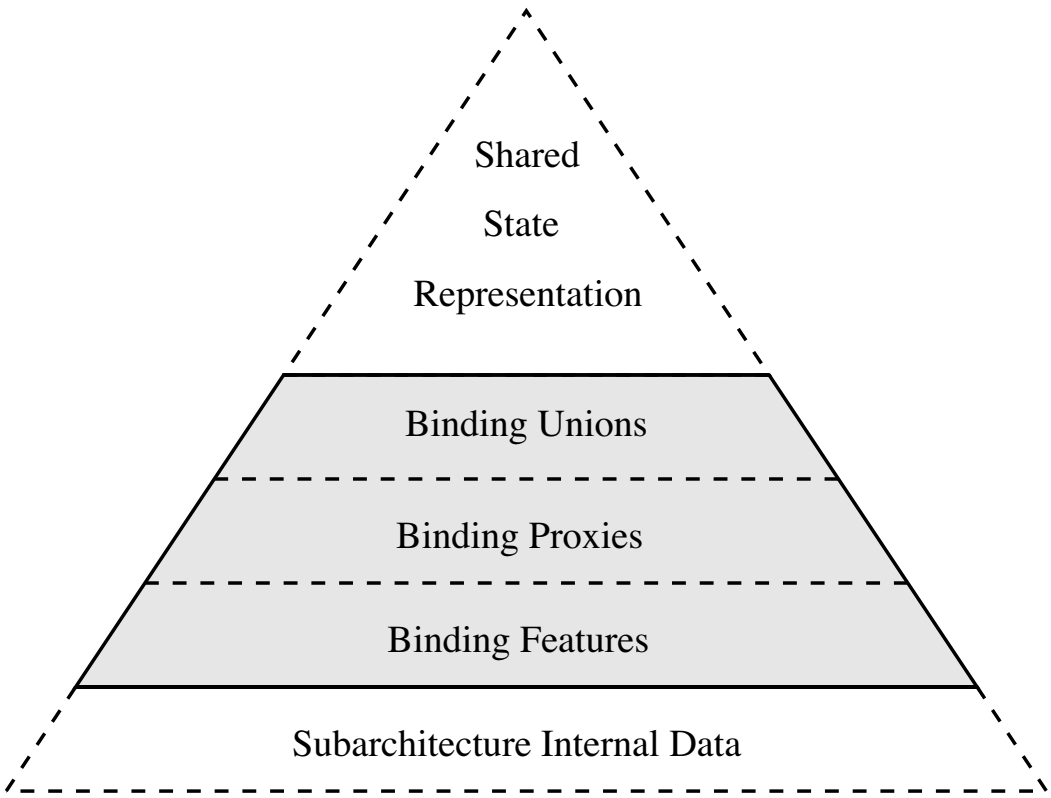
Requirements
The Relative Intrusiveness of Comparators

- If a new feature is added, a function which compare that type feature to other comparable types *should* also be added
  - These functions are called *comparators*
  - In current implementation, they should return *true*, *false* or *indeterminate* for every pair of feature instances (brutally simple)
- The result from the comparisons is the basis for the *binding score* which in turn decides which proxies may in fact refer to the same entity
- Comparators may be based on anything, e.g.
  - hardcoded knowledge (e.g. equivalence testing)
  - ontological reasoning
  - learned mappings
  - context aware agents
  - etc.

Requirements
3. Stability of Symbols

- A proxy is precisely a ... *proxy* for an entity
- A subarchitecture which creates a proxy will use that proxy as an internal symbol for the represented entity
- The proxy is constant w.r.t the subarchitecture
- Based on the binding score, proxies are unified into *binding unions*
- Unions provide an enriched description of the proxies
- As proxies are added, the existing unions are scored and “compete” to bind
- Unions change frequently, proxies are stable

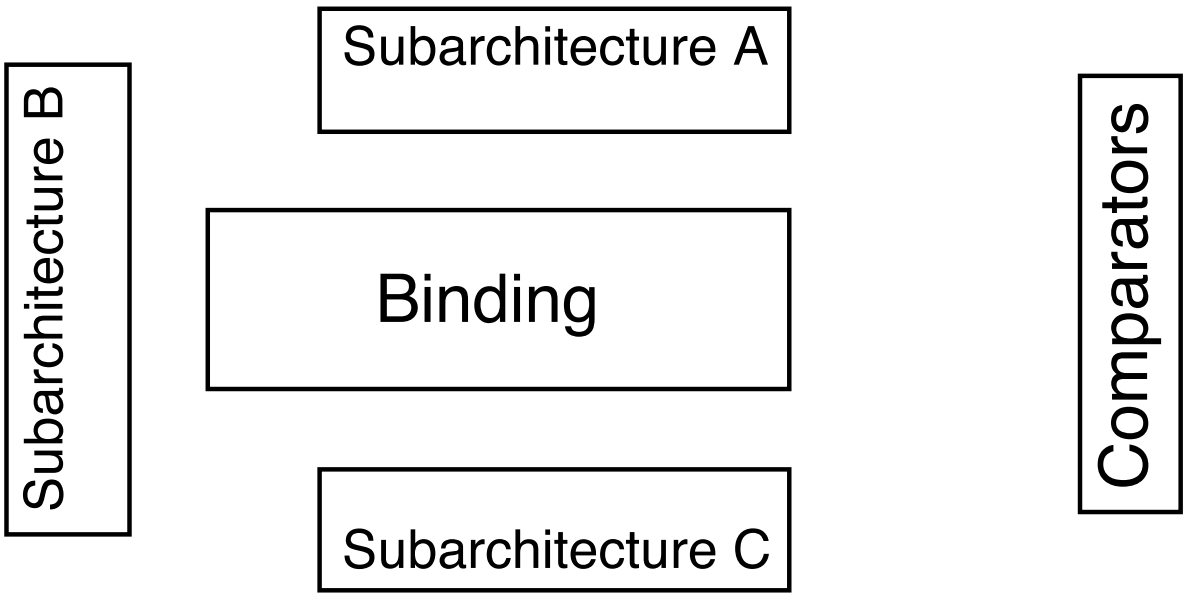
Representation Summary



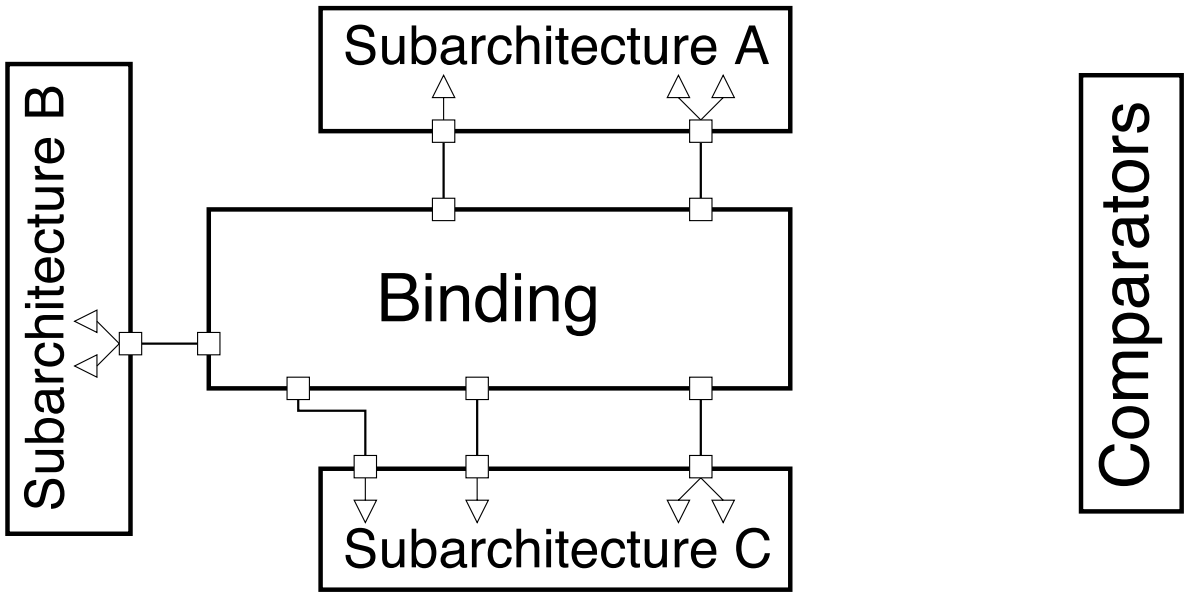
The Binder

- The *binder* is a subarchitecture among the others in CAST
- Tasks:
  - Invoke comparators
  - Calculate the binding scores (unions vs. proxies)
  - Create unions
  - Identify disambiguation issues
  - Signal subarchitectures whenever their proxies are bound/rebound
  - Administration...
- The binder does this without a clue about what is represented
- Asynchronous additions and updates of proxies is handled

Binder

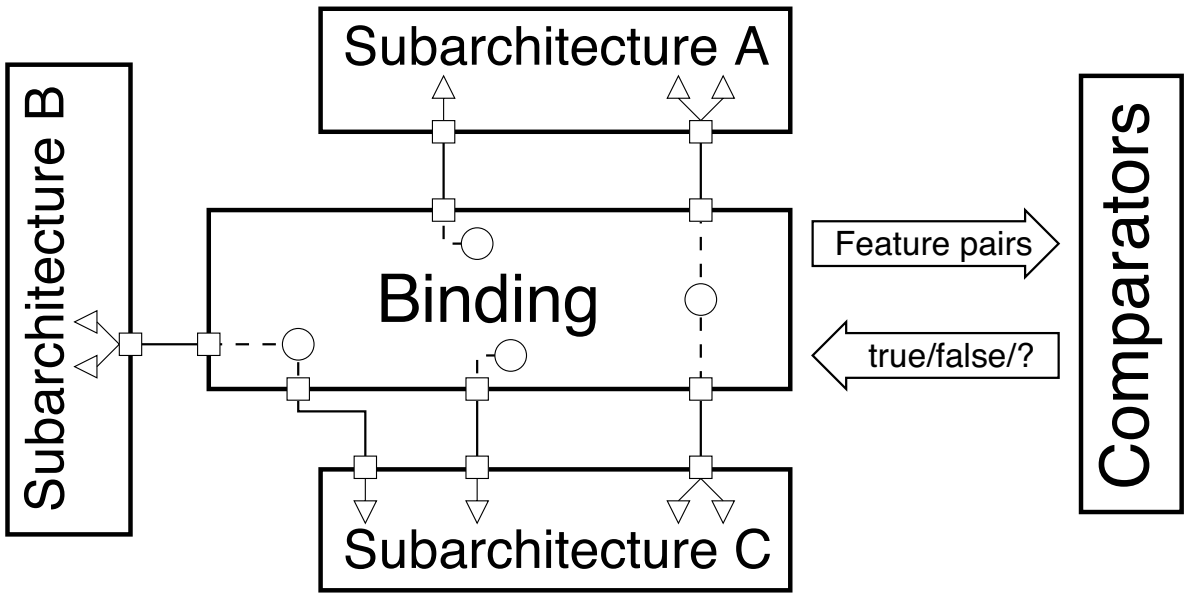


Binder

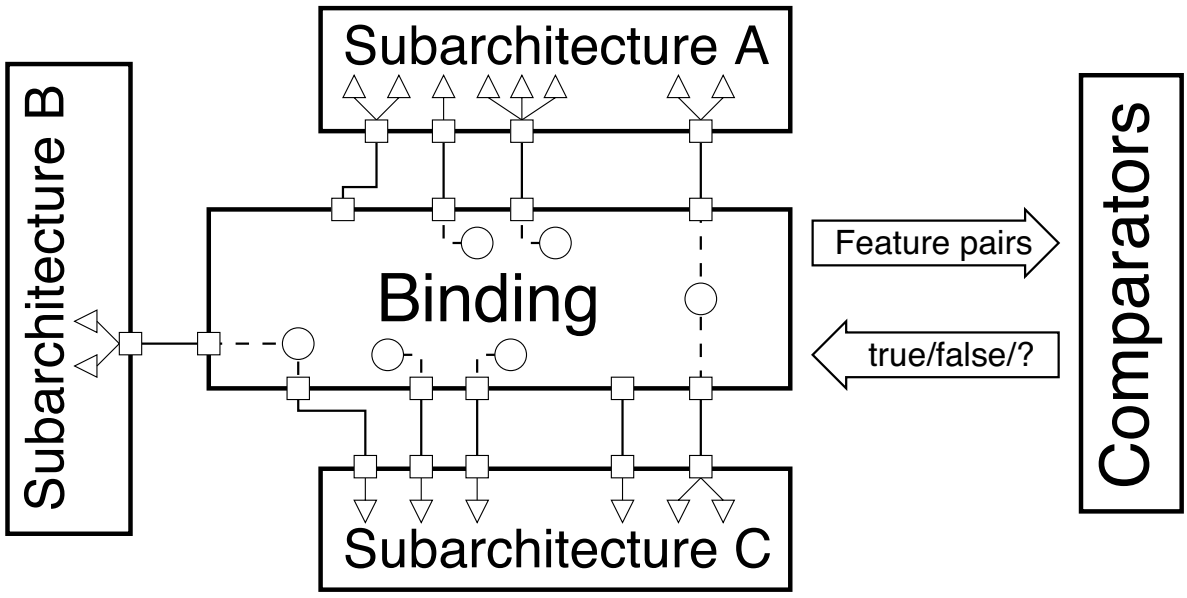


Binder

Binding
Jacobsson, Hawes, Kruijff, Wyatt
Introduction
The Problem
Motivation
The Binder
Discussion

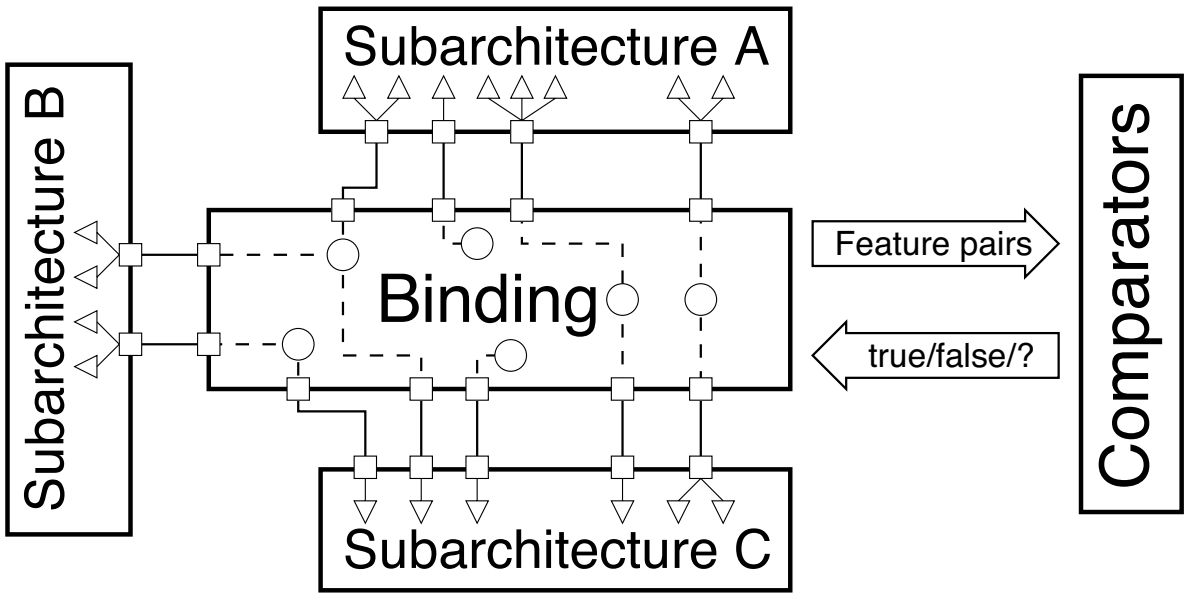


Binder



Binder

Binding
Jacobsson, Hawes, Kruijff, Wyatt
Introduction
The Problem
Motivation
The Binder
Discussion



How We Use/Intend To Use the Binder

- Primary clients
  - Communication
  - Planning
- Scenarios
  - Tabletop scenarios
  - Human augmented mapping
  - Incremental processing (subarchitectures can serve as a source of heuristics for each other)
  - Tutoring scenarios

Some Positive Consequences

- Representational freedom
- Disambiguation issues identified
- Comparators are implemented by experts
- Modal and amodal representations side by side
- Subsymbolic representation make varying abstraction possible
- Lazy binding
- Incrementality and asynchronous processing
- Scalable
- Small demands on subarchitectures

Less Positive Consequences

- Information fusion aspect is very limited since the binding score is extremely simple (no fuzzy or Bayesian scoring etc)
  - internally, comparators may be as SOTA as they like, though
- Lack of comparable features can be problematic
- Anything can be a binding feature, not everything should though
- Anyone can propose proxies as they like, but can you trust everyone?
- It's important that subarchitectures are conservative about proposing proxies! (layered binding an option otherwise)

Future Work

- Incorporate other approaches to symbol grounding as subcomponents (as comparators or monitors)
- Enrich the binding score to accomodate representations of belief of comparators
- Episodic memory