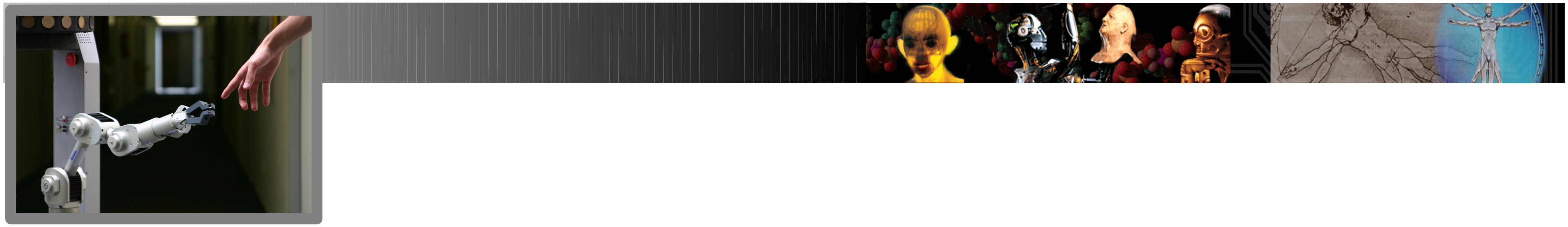


A new design for the CogX binder

Draft (version 0.5)
comments welcome !

Pierre Lison
[pierre.lison@dfki.de]

Language Technology Lab
DFKI GmbH, Saarbrücken
http://talkingrobots.dfki.de

Intro

- The present document is a first draft detailing the new binding approach that we are currently developing for CogX
- Many ideas presented here are still very tentative
  - Comments, criticisms, ideas, remarks, suggestions are *most* welcome!
  - Sentences coloured in green indicate open questions or unresolved issues
- In terms of implementation, a first release with a core binder is expected for *mid-July*, a second, extended release for *September*, and a third one for *after the review meeting*
  - Needless to say, if you're interested in this approach and would like to be involved in the design or development of this binder, you're *most* welcome as well :-)

Outline

(1) Why a new design?

(2) A Bayesian approach to binding

(3) Functionality requirements

(4) Design sketch

(5) Notes & open questions

Outline

(1) Why a new design?

(2) A Bayesian approach to binding

(3) Functionality requirements

(4) Design sketch

(5) Notes & open questions

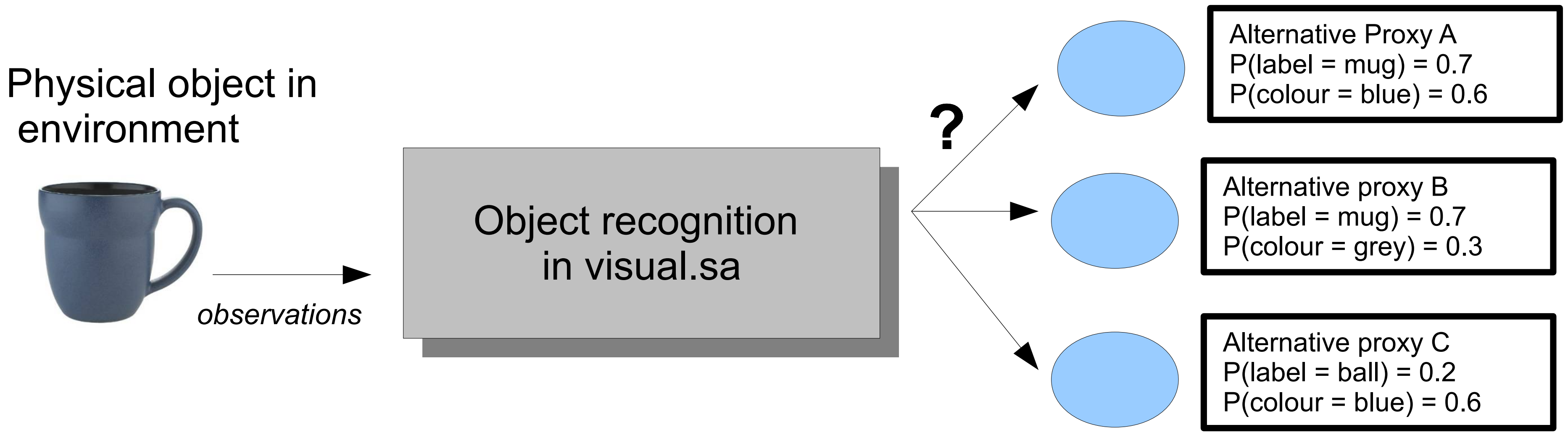
Why a new design?

- Why redesign and reimplement the binder from scratch, instead of reusing the old one?
- **To put it simply**: the current binder design doesn't really scale up to address the scientific and technical issues of the CogX project
- “Weak spots” in the current designs:
  - Current design implicitly assumes *full observability* of the current environment (i.e., assuming no noise and no uncertainty in the measurements, and no inaccuracies/errors in the modal interpretations)
  - Current feature comparators can express whether two feature instances are compatible, incompatible, or indifferent, but are unable to express fine-grained *correlations* between them
  - *Conceptual confusion* on the nature of the binding proxies
  - Implicit *learning* and *adaptation* are difficult to cast in the design; temporal *instability*, lack of *robustness* in presence of noisy data
  - Various technical, implementation-specific problems

Why new design? Uncertainty

1. Current design implicitly assumes *full observability* of the current environment
   - In other words, it does not account for the various levels of **(un)certainty** which are associated with modality-specific observations.

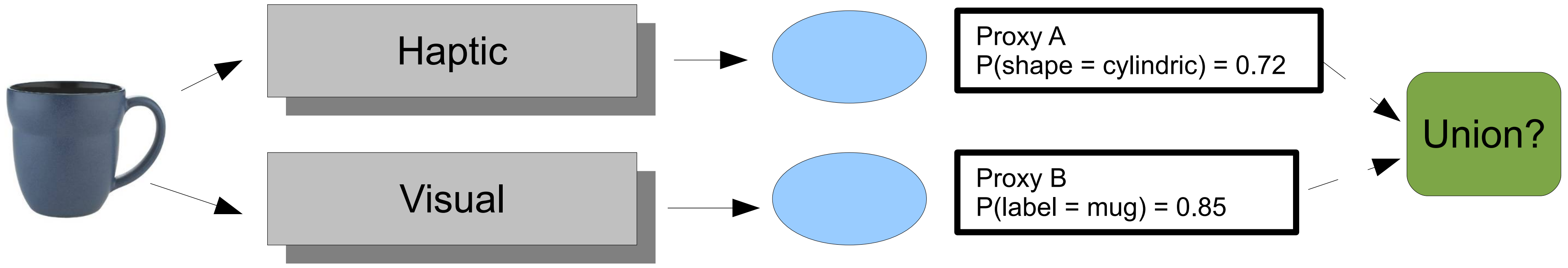
     → We would like to integrate information about the **probability** of the each given observation at the core of the binding process
   - *Example*: for a given object, the visual subarchitecture should output not a single, but a set of alternative recognition hypotheses, each having its own probability (joint probability of its features)



Why new design? Comparators

2. Current feature comparators can only indicate whether two feature instances are compatible, incompatible, or indifferent ("indeterminate"), but nothing else

   - We'd like to express more fine-grained statements, indicating various levels of **correlation** between pairs of feature instances
   - Example: a feature instance "*shape: cylindric*" produced by a haptic subarchitecture might correlate more or less strongly with a "*label: mug*" feature instance from the object recognition
   - These correlations should also be applicable on *continuous features*
   - The should be gradually *learned* by the cognitive agent



Why new design? Semantics issues

3. The old design introduced some conceptual confusion on the exact *nature* of the binding proxies

- What does it mean for something to be "on the binder"? What is precisely the semantics of the representations manipulated by the binder?
- No explicit distinction between the "perceptive" proxies and the "linguistic" proxies... is that adequate?

• **Perceptive proxies** (produced by the visual, spatial, haptic, navigation subarchitectures) all ultimately arise from *sensory input*. • Perceptive proxies therefore always describe some (objective) aspects of the *current situated reality.* Their spatio-temporal frame is fixed in advance.	•**Linguistic proxies**, on the other hand, can freely refer to the past, present, future, imaginary or hypothetical worlds, counterfactuals, and events/objects in remote places. • Linguistic proxies can therefore choose (via linguistic cues) the *spatio-temporal frame* in which they are to be evaluated.

Why new design? Scaling problems

4. No easy way to specify correlations between sets of feature instances beyond pairs (for instance, correlations between 3 or 4 co-occurring features)
5. Difficulties dealing with temporally unstable proxies (visual proxies, for instance)
6. Implicit learning and adaptation are difficult to cast in the old design (at least if we venture outside highly artificial setups)
7. Interfacing binder content with a decision-theoretic planner (like the one developed by Richard) not straightforward
8. Lack of robustness in presence of noisy data
9. Binding ambiguities must be explicitly encoded in the WMs, and treated *ad hoc*

Why a new design? Implementation

10. Various implementation-specific problems:

- Difficult maintainability/portability (code size, readability etc.);
- Poor efficiency (runtime speed, memory use);
- Runtime instability (due to the intense use of multithreading) ;
- Binding interface has become very complex over time, and hence difficult to understand and use
- Presence of various programming workarounds and idiosyncrasies which are no longer useful

Outline

(1) Why a new design?

(2) A Bayesian approach to binding

- Proxies and unions
- Framing the binding problem
- How to estimate the parameters
- The Binding Algorithm
- Probabilistic reasoning over time

(3) Functionality requirements

(4) Design sketch

(5) Notes & open questions

The new approach to binding

- We propose a new binder design which seek to address (at least partially) these issues

- **Bayesian** approach to binding:
    - Takes into account the *uncertainty* of the data contained in the proxies
    - Increased *robustness* and *flexibility*
    - Possibility of specifying various kinds of *correlation* between features
    - *Learning* and *adaptation* at the core of the design
    - Strong *theoretical foundations* in probability theory

Outline

(1) Why a new design?

(2) A Bayesian approach to binding

- **Proxies and unions**
- Framing the binding problem
- How to estimate the parameters
- The Binding Algorithm
- Probabilistic reasoning over time

(3) Functionality requirements

(4) Design sketch

(5) Notes & open questions

Proxies and unions

- The two central data structures manipulated by the binder are **proxies** and **unions**
- A **<u>proxy</u>** is an abstract representation of a given entity in the environment, as perceived by a particular modality
  - A proxy is defined as a list of <feature, feature value> pairs
    - Each of these pairs is associated to a probability
    - Features values can be discrete or continuous
    - Two feature pairs of particular importance are the *saliency* of the entity, and its *spatio-temporal frame* (cf. gj's formalization of these notions)
    - The existence of the proxy itself is also associated to a probability (cf. next slide)
- An **<u>union</u>** is an abstract representation of a given entity in the environment, which combines the information perceived by one or more modalities, *via* the proxies generated by the subarchitectures
  - It is also defined as a list of <feature, feature value> pairs with associated probabilities, plus references to the proxies included in the union

The proxy data structure

Example of a specific proxy *a*, derived from the observation vector **z** coming from the sensors:

Proxy a:
- Exists with prob. 0.91
- Shape = *cylindric* with prob. 0.87
- Colour = *red* with prob. 0.94

Or equivalently, using the standard probabilistic notation:

Proxy a:

$P(\text{Exists(a)=T}|\ \mathbf{z}) = 0.91$

$P(Shape(a)\text{=cylindric}\ |\ \text{Exists(a)=T},\ \mathbf{z}) = 0.87$

$P(Colour(a)\text{=red}\ |\ \text{Exists(a)=T},\ \mathbf{z}) = 0.94$

- $P(\text{Exists(a)=T}|\ \mathbf{z})$ is the probability that, given the observation z, the physical entity associated with the proxy *a* exists in the environment
  - In other words, it is the probability that the proxy correctly match an entity in the real world (and was not thus created *ex nihilo* by the subarchitecture, because e.g of the noise or an interpretation error)
- The two other probabilities express the probability that the perceived properties of shape and colour match the real properties of the entity

Proxies as multivariate probabilistic distributions

- What happens if, instead of outputting a single proxy hypothesis for a given entity, the subarchitecture outputs a set of alternative hypotheses?
- For instance, we might have for the same proxy:
  - Shape = *cylindric* with prob. 0.87
  - Shape = *spherical* with prob. 0.11
- Explicitly producing distinct proxies for every single hypothesis is inelegant and largely inefficient. We need a way to easily define alternative hypotheses for a given proxy
  - To this end, we define a proxy as specifying a **multivariate probabilistic distribution** over possible feature values
  - This distribution effectively specifies all possible alternative hypotheses for a given proxy

Proxies as multivariate probabilistic distributions

- The proxy *a* represents a multivariate probability distribution over the *n*-dimensional random variable $\mathbf{f_a} = (f_1(a), f_2(a), ...f_n(a))$

- This distribution is technically defined via the following probability mass function (pmf) over $\mathbf{x}$:

$$\begin{aligned} f_{\mathbf{f_a}}(\mathbf{x}) &= P(\mathbf{f_a} = \mathbf{x} \mid \mathbf{z}) \\ &= P(f_1(a){=}x_1, f_2(a){=}x_2, ...f_n(a){=}x_n \mid \text{Exists(a)=T},\ \mathbf{z}) \\ &\quad \times P(\text{Exists(a)=T} \mid \mathbf{z}) \end{aligned}$$

- Under the assumption of independence of the feature probabilities *given the observation of the object*, this pmf becomes

$$\approx \left[\prod_{i=1}^{n} P(f_i(a){=}x_i \mid \text{Exists(a)=T}, \mathbf{z}) \right] \ P(\text{Exists(a)=T} \mid \mathbf{z})$$

Proxies as multivariate probabilistic distributions

- For the distribution to be well-defined over all dimensions 1...n, we need to ensure that

$$\sum_u P(f_i{=}u) = 1$$

(with *u* running through the set of all possible values of f_i)

is satisfied for all dimensions $1 \leq i \leq n$.

- This can be easily enforced by allowing a new feature value *indeterminate* to be assigned for all features
    - The feature value *indeterminate* ensures that the sum of the probabilities of all features values for a feature f_i equals to 1.
    - For instance,

$$P(Shape(a){=}\text{indeterminate} \mid \text{Exists(a)=T},\ \mathbf{z}) = 0.02$$

Proxies as multivariate probabilistic distributions

Example of a proxy with 2x2 alternative hypotheses, derived from the observations vector **z** coming from the sensors:

Proxy a:
- Exists with prob. 0.91
- Shape = *cylindric* with prob. 0.87
- Shape = *spherical* with prob. 0.11
- Colour = *red* with prob. 0.94
- Colour = *blue* with prob. 0.02

Or equivalently, using the standard probabilistic notation:

Proxy a:

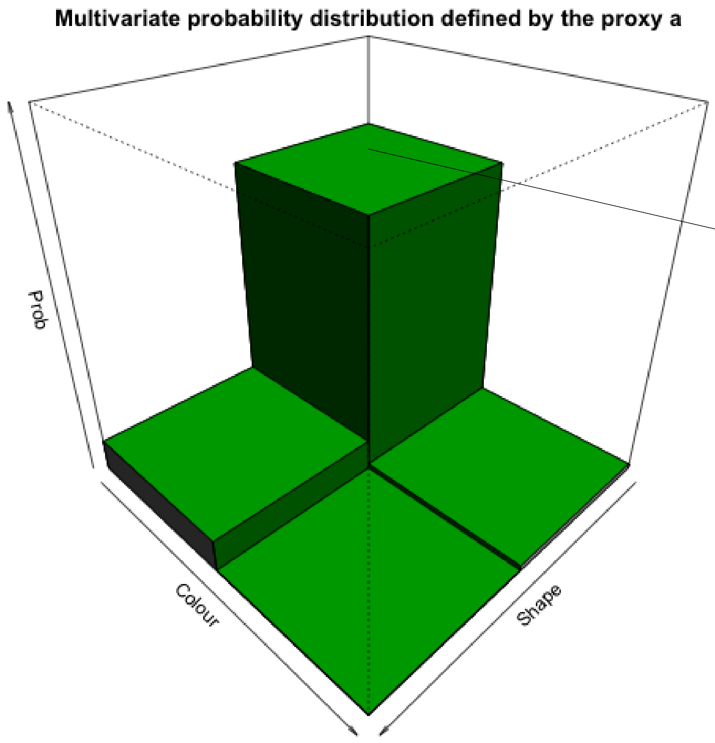
$$P(\text{Exists(a)=T}|\ \mathbf{z}) = 0.91$$
$$P(Shape(a)\text{=cylindric} \mid \text{Exists(a)=T},\ \mathbf{z}) = 0.87$$
$$P(Shape(a)\text{=spherical} \mid \text{Exists(a)=T},\ \mathbf{z}) = 0.11$$
$$P(Colour(a)\text{=red} \mid \text{Exists(a)=T},\ \mathbf{z}) = 0.94$$
$$P(Colour(a)\text{=blue} \mid \text{Exists(a)=T},\ \mathbf{z}) = 0.02$$

- The 2-dimensional probability mass function defined by the proxy is

$$\begin{aligned} P\left((Colour(a), Shape(a)) = (x_1, x_2) \mid \mathbf{z}\right) = {} & P(Colour(a)\text{=}x_1 \mid \text{Exists(a)=T},\ \mathbf{z}) \\ & \times P(Shape(a)\text{=}x_2 \mid \text{Exists(a)=T},\ \mathbf{z}) \\ & \times P(\text{Exists(a)=T} \mid \mathbf{z}) \end{aligned}$$

Proxy distributions, visually

- For our 2-dimensional example, the probabilistic distribution defined by the proxy *a* can be plotted as such:



$$P\left(Shape(a){=}\text{cylindric}, Colour(a){=}\text{red} \mid \mathbf{z}\right) =$$

$$P(Shape(a){=}\text{cylindric} \mid \text{Exists(a)=T},\ \mathbf{z})$$
$$\times P(Colour(a){=}\text{red} \mid \text{Exists(a)=T},\ \mathbf{z})$$
$$\times P(\text{Exists(a)=T} \mid \mathbf{z})$$

$$= 0.7442$$

- Of course, most proxies will have more than two features, resulting in a *n*-dimensional probability distribution

Using continuous features

- All the features introduced until now were *discrete* features, taking their feature values in a *finite* domain (the set of possible shapes, for instance).
- But many features of the robot's environment are *continuous* rather than discrete:
  - Examples: the size of the object, the metric location of a room, the saliency of an object, etc.
- We would like to be able to define continuous features as well as discrete features in the binding proxies.
  - But while keeping the whole probabilistic framework!

Continuous features

- To this end, we define the feature value of a continuous feature using a **probability density function** (pdf)
  - Of course, we don't need to define the complete pdf formula in detail – we just specify the *type* and *parameters* of the pdf
  - For instance, if we assume the exact size of an object to be normally distributed with mean μ = 32.4 cm, and variance σ^2 = 1.1, we specify the feature value as **size = *N* (32.4, 1.1)**
  - The resulting pdf is $p(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$ with μ = 32.4, σ^2 = 1.1
- The probability of having the object size comprised between two particular values (say, between 30 and 34 cm) can then be derived by integrating the pdf over these two values:

$$P(30 \leq size \leq 34 \mid \mathbf{z}) = \int_{30}^{34} p(x)$$

Using continuous features (2)

Example of a proxy with alternative hypotheses and continuous features, derived from the observations vector **z** coming from the sensors:

Proxy a:

- Exists with prob. 0.91
- Shape = *cylindric* with prob. 0.87
- Shape = *spherical* with prob. 0.11
- Colour = *red* with prob. 0.94
- Colour = *blue* with prob. 0.02
- Size = (gaussian with mean = 32.4 and variance = 1.1)

Or equivalently, using the standard probabilistic notation:

Proxy a:

$$P(\text{Exists(a)=T}|\ \mathbf{z}) = 0.91$$

$$P(Shape(a)\text{=cylindric} \mid \text{Exists(a)=T},\ \mathbf{z}) = 0.87$$

$$P(Shape(a)\text{=spherical} \mid \text{Exists(a)=T},\ \mathbf{z}) = 0.11$$

$$P(Colour(a)\text{=red} \mid \text{Exists(a)=T},\ \mathbf{z}) = 0.94$$

$$P(Colour(a)\text{=blue} \mid \text{Exists(a)=T},\ \mathbf{z}) = 0.02$$

$$P(\text{a} \leq Size(a) \leq \text{b} \ |\text{Exists(a)=T},\ \mathbf{z}) = \int_a^b N(32.4, 1.1)$$

- **Q1**: How to formally express the complete mixed continuous/discrete probabilistic distribution?

Relation proxies

- Relation proxies are a particular type of proxies which can be generated within a subarchitecture S_i
- As proxies, they are also defined as a list of features with probabilities
- But they also have two additional specific features: *source_proxy* and *target_proxy*
  - The domain of these two features is the set of proxies generated within S_i (excluding the relation proxy itself, of course)
  - The values taken by *source_proxy* and *target_proxy* must be distinct

Proxy b:
• <u>Exists</u> with prob. 0.87
• + List of features...

Proxy a:
• <u>Exists</u> with prob. 0.91
• + List of features...

Proxy r:
• <u>Exists</u> with prob. 0.66

• <u>Source_proxy</u> = a with prob. 0.89
• <u>Target_proxy</u> = b with prob. 0.98

• + List of features...

Binding unions

- Unions are basically described in the same way as proxies
  - That is, as lists of <*feature, feature value*> pairs with probabilities attached to each of them.
- The list of features instances in a binding union is the union of the list of the included proxies
  - i.e. If two proxies *A* and *B* merge to form an union *U*, the feature list of *U* is simply defined as:

$$\mathbf{f}_U = \mathbf{f}_A \cup \mathbf{f}_B$$

- If features from different proxies have the *exact* same label, then their feature values are merged in a *single feature instance* in the union
  - If the features have discrete values, their respective values must coincide
  - The resulting probability of this merged feature will be detailed in the next slides

Outline

(1) Why a new design?

(2) A Bayesian approach to binding

- Proxies and unions
- **Framing the binding problem**
- How to estimate the parameters
- The Binding Algorithm
- Probabilistic reasoning over time

(3) Functionality requirements

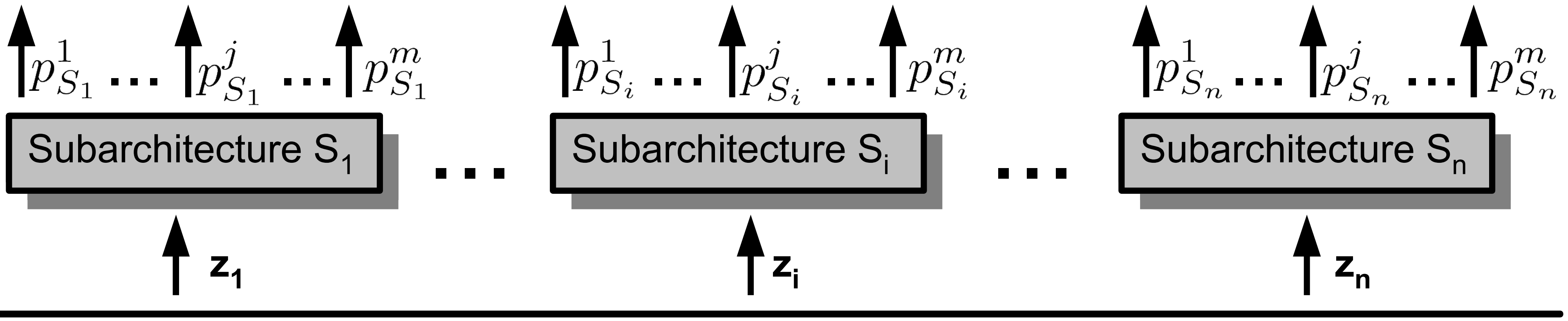
(4) Design sketch

(5) Notes & open questions

The binding problem

- Consider the following
  - A set of subarchitectures $S_1 ... S_n$
  - Each subarchitecture S_i perceives an observation vector $\mathbf{z_i}$
  - Based on its observation vector $\mathbf{z_i}$, the subarchitecture S_i generates a set of proxies $= \{p^j_{S_i} | 1 \leq j \leq m\}$

    (the nb. of generated proxies *m* can be different for each subarchitecture)
  - Each proxy $p^j_{S_i}$ is composed of a feature vector $\mathbf{f_{p^j_{S_i}}} = (f^1_{p^j_{S_i}} ... f^o_{p^j_{S_i}})$
  - The possible values taken by the proxy for each of these features are defined via the multivariate probability distribution $P(\mathbf{f_{p^j_{S_i}}} = \mathbf{x} \mid \mathbf{z_i})$

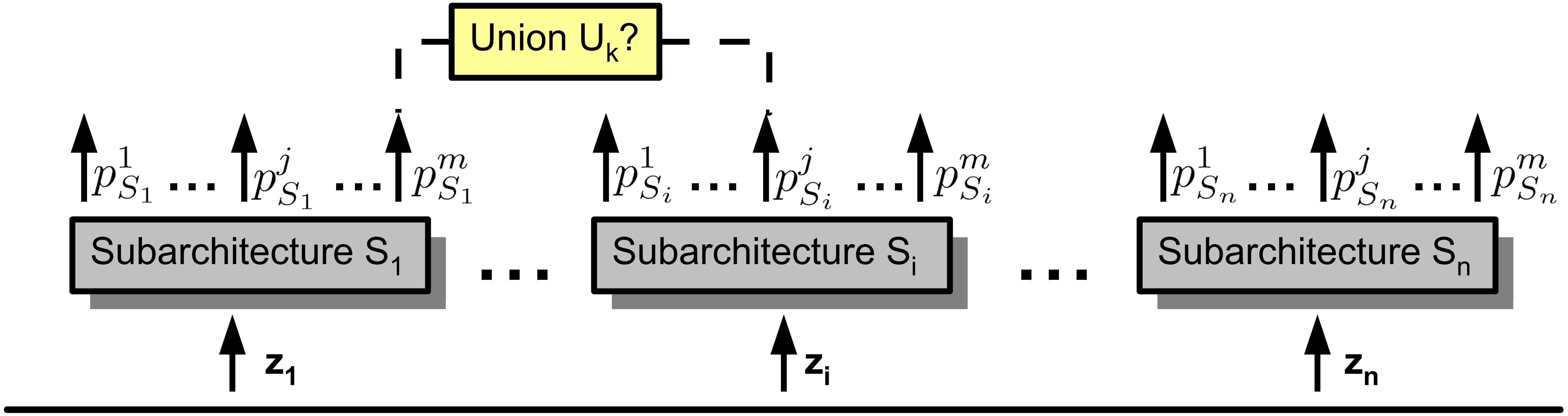


$p^1_{S_1}$ ... $p^j_{S_1}$ ... $p^m_{S_1}$
$p^1_{S_i}$ ... $p^j_{S_i}$ ... $p^m_{S_i}$
$p^1_{S_n}$ ... $p^j_{S_n}$ ... $p^m_{S_n}$
Subarchitecture S_1
...
Subarchitecture S_i
...
Subarchitecture S_n
$\mathbf{z_1}$
$\mathbf{z_i}$
$\mathbf{z_n}$

The binding problem (2)

- The task to solve:
  - Given these generated proxies, we are looking for possible *binding unions* over them (with at most one proxy per subarchitecture)
  - Let U_k be a possible union over a set of proxies
  - This union is composed of a feature vector $\mathbf{f_{U_k}} = (f^1_{U_k} \dots f^q_{U_k})$
  - To decide whether or not U_k is a good candidate to form an union, we have to compute its probability, given all the observations that we have:

$$P(\mathbf{f_{U_k}} \mid \mathbf{z_1}, \dots \mathbf{z_n})$$



Multimodal data fusion

- To determine the probability of a given union, we have to combine probabilistic information arising from several information sources (the observation vectors $\mathbf{z_1} ... \mathbf{z_n}$):

$$P(\mathbf{f_{U_k}} \mid \mathbf{z_1}, ...\mathbf{z_n})$$

- Several techniques have been developed in the multi-modal data fusion community to address this problem
- In our design, we will use a well-known technique called **Independent Likelihood Pool** (ILP)
- The ILP is founded on the usual Bayes rule:

$$P(\mathbf{f_{U_k}} \mid \mathbf{z_1}, ...\mathbf{z_n}) = \frac{P(\mathbf{z_1}, ...\mathbf{z_n} \mid \mathbf{f_{U_k}}) \; P(\mathbf{f_{U_k}})}{P(\mathbf{z_1}, ...\mathbf{z_n})}$$

Independent Likelihood Pool

- The main assumption behind ILP is that the information sources $\mathbf{z_1} \ldots \mathbf{z_n}$ can be seen as independent from one another *given* $\mathbf{f_{U_k}}$
  - this is a reasonable assumption, since the only parameter that these observations have in common is precisely the physical entity represented by U_k which generated these observations!
- We can therefore write:

$$P(\mathbf{z_1} \ldots \mathbf{z_n} \mid \mathbf{f_{U_k}}) = \prod_{i=1}^{n} P(\mathbf{z_i} \mid \mathbf{f_{U_k}})$$

- Inserting this into the previous formula:

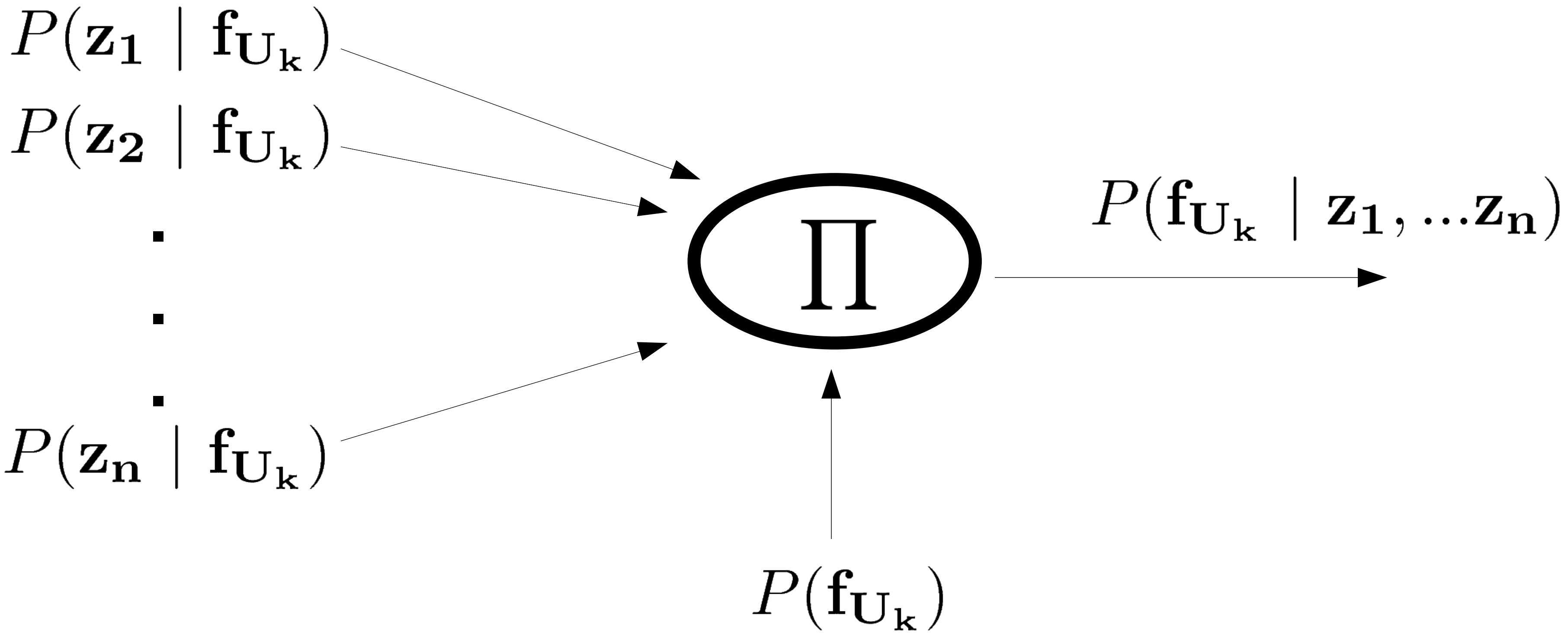
$$P(\mathbf{f_{U_k}} \mid \mathbf{z_1}, \ldots \mathbf{z_n}) = \frac{P(\mathbf{z_1}, \ldots \mathbf{z_n} \mid \mathbf{f_{U_k}})\ P(\mathbf{f_{U_k}})}{P(\mathbf{z_1}, \ldots \mathbf{z_n})}$$

$$= \alpha\ P(\mathbf{f_{U_k}}) \prod_{i=1}^{n} P(\mathbf{z_i} \mid \mathbf{f_{U_k}})$$

normalisation constant

The ILP, graphically

$$P(\mathbf{f_{U_k}} \mid \mathbf{z_1}, \ldots \mathbf{z_n}) = \alpha \, P(\mathbf{f_{U_k}}) \prod_{i=1}^{n} P(\mathbf{z_i} \mid \mathbf{f_{U_k}})$$


$P(\mathbf{z_1} \mid \mathbf{f_{U_k}})$
$P(\mathbf{z_2} \mid \mathbf{f_{U_k}})$
$P(\mathbf{z_n} \mid \mathbf{f_{U_k}})$
$\prod$
$P(\mathbf{f_{U_k}} \mid \mathbf{z_1}, \ldots \mathbf{z_n})$
$P(\mathbf{f_{U_k}})$

Taking stock

- **What we seek:**
  - We want to compute the probability $P(\mathbf{f_{U_k}} \mid \mathbf{z_1}, ...\mathbf{z_n})$ of a given union given the modal observation vectors $\mathbf{z_1} ... \mathbf{z_n}$
  - The *Independent Likelihood Pool* allows us to rewrite the probability:

$$P(\mathbf{f_{U_k}} \mid \mathbf{z_1}, ...\mathbf{z_n}) = \alpha\ P(\mathbf{f_{U_k}}) \prod_{i=1}^{n} P(\mathbf{z_i} \mid \mathbf{f_{U_k}})$$

  - But how do we compute the conditional probabilities $P(\mathbf{z_i} \mid \mathbf{f_{U_k}})$?

- **What we have:**
  - The proxies generated by the subarchitectures
  - Each proxy $p^j_{S_i}$ (with $1 \leq i \leq n$ and $1 \leq j \leq m$) defines a multivariate probability distribution:

$$P(\mathbf{f_{p^j_{S_i}}} = \mathbf{x} \mid \mathbf{z_i}) = P(f^1_{p^j_{S_1}} = x_1, f^2_{p^j_{S_1}} = x_2, ...f^o_{p^j_{S_1}} = x_o \mid \mathbf{z_i})$$

Rewriting the ILP

- Let us see how we can rewrite the formula given by the ILP in order to use the probability distributions specified by the proxies
- We can distinguish two possible cases:

1. The binding union U_k does *not* include any proxy generated by S_i
   - The probability $P(\mathbf{z_i} \mid \mathbf{f_{U_k}})$ can then be simplified to $P(\mathbf{z_i})$
2. The binding union U_k *does* include a proxy $p^j_{S_i}$ generated by S_i

   (NB: remember that by definition, an union can only include *at most* one proxy per subarchitecture)
   - The conditional independence assumption does not hold anymore
   - But we can rewrite $P(\mathbf{z_i} \mid \mathbf{f_{U_k}})$ in terms of the probability of the proxy
   - After some rewriting steps, this gives us:

$$P(\mathbf{z_i} \mid \mathbf{f_{U_k}}) = P(\mathbf{z_i} \mid \mathbf{f_{p^j_{S_i}}})\ P(\mathbf{f_{p^j_{S_i}}} \mid \mathbf{f_{U_k}})$$

(this also assumes that, apart from the proxy $p^j_{S_i}$, the probability of the observation vector $\mathbf{z_i}$ is independent from all other proxies from S_i)

Rewriting the ILP (2)

- The formula we have now is

$$P(\mathbf{z_i} \mid \mathbf{f_{U_k}}) = P(\mathbf{z_i} \mid \mathbf{f_{p^j_{S_i}}})\ P(\mathbf{f_{p^j_{S_i}}} \mid \mathbf{f_{U_k}})$$

- It can be further simplified
  - Using Bayes rule:

$$P(\mathbf{z_i} \mid \mathbf{f_{p^j_{S_i}}}) = \frac{P(\mathbf{f_{p^j_{S_i}}} \mid \mathbf{z_i})\ P(\mathbf{z_i})}{P(\mathbf{f_{p^j_{S_i}}})}$$

  - And $P(\mathbf{f_{p^j_{S_i}}} \mid \mathbf{f_{U_k}})$ is simply equal to … 1!

    (*Why?* The union U$_k$ includes all the information contained in the proxy, so given the union, the probability of the proxy is equal to 1)

- This finally gives us:

$$P(\mathbf{z_i} \mid \mathbf{f_{U_k}}) = \frac{P(\mathbf{f_{p^j_{S_i}}} \mid \mathbf{z_i})\ P(\mathbf{z_i})}{P(\mathbf{f_{p^j_{S_i}}})}$$

In other words: Given that you see a red apple, you can be sure that the probability of seeing an apple = 1.0 !

Rewriting the ILP (3)

- Getting back to the initial ILP formula:

$$P(\mathbf{f_{U_k}} \mid \mathbf{z_1}, ...\mathbf{z_n}) = \alpha \; P(\mathbf{f_{U_k}}) \prod_{i=1}^{n} P(\mathbf{z_i} \mid \mathbf{f_{U_k}})$$

$$= \alpha \; P(\mathbf{f_{U_k}}) \times \prod_{\{i : \nexists p^j_{S_i} \in \text{ proxies}(U_k)\}} P(\mathbf{z_i})$$

$$\times \prod_{\{i : \exists p^j_{S_i} \in \text{ proxies}(U_k)\}} \frac{P(\mathbf{f_{p^j_{S_i}}} \mid \mathbf{z_i}) \; P(\mathbf{z_i})}{P(\mathbf{f_{p^j_{S_i}}})}$$

- And simplifying again:

$$P(\mathbf{f_{U_k}} \mid \mathbf{z_1}, ...\mathbf{z_n}) = \beta \; P(\mathbf{f_{U_k}}) \times \prod_{\{p^j_{S_i} \in \text{proxies}(U_k)\}} \frac{P(\mathbf{f_{p^j_{S_i}}} \mid \mathbf{z_i})}{P(\mathbf{f_{p^j_{S_i}}})}$$

The binding formula

- The final binding formula we derived:

$$P(\mathbf{f_{U_k}} \mid \mathbf{z_1}, \ldots \mathbf{z_n}) = \beta \; P(\mathbf{f_{U_k}}) \times \prod_{\{p^j_{S_i} \in \text{proxies}(U_k)\}} \frac{P(\mathbf{f_{p^j_{S_i}}} \mid \mathbf{z_i})}{P(\mathbf{f_{p^j_{S_i}}})}$$

Normalisation constant

Number of included proxies

Probability of the union given the evidence of all modalities

Prior probability of the union (“internal consistency“)

Prior probability of the proxy

Probability of the proxy given the evidence from z_i

Additional notes

- We could / should maybe introduce a slack variable in the binding formula we just presented. This slack variable could be used to dynamically control the greediness of the binder
- **Q2**: Once we get the probability of the whole union, how do we efficiently compute the probability of each feature individually?
  - By integrating out all the features except one, and repeat this procedure for every feature in the union?
  - Or simpler: just take the feature probability specified in the included proxies

Outline

(1) Why a new design?

(2) A Bayesian approach to binding

- Proxies and unions
- Framing the binding problem
- **How to estimate the parameters**
- The Binding Algorithm
- Probabilistic reasoning over time

(3) Functionality requirements

(4) Design sketch

(5) Notes & open questions

Ingredients for the binding formula

- To derive the probability of a given union, we therefore need to compute the following values:
  - The prior probability of the union $\boxed{P(\mathbf{f}_{\mathbf{U_k}})}$, which serves as a measure of the "internal consistency" of the union
  - And for each proxy $p^j_{S_i}$ included in the union:
    - The probability of the proxy given the modality-specific observation vector: $\boxed{P(\mathbf{f}_{\mathbf{p^j_{S_i}}} \mid \mathbf{z_i})}$
    - And the prior probability of the proxy $\boxed{P(\mathbf{f}_{\mathbf{p^j_{S_i}}})}$
- How do we estimate these values?
  - Or in other words: *where do we get the probabilities from*?

Outline

(1) Why a new design?

(2) A Bayesian approach to binding

- Proxies and unions
- Framing the binding problem
- How to estimate the parameters
  - **Prior union probability**
  - Proxy probabilities
- The Binding Algorithm
- Probabilistic reasoning over time

(3) Functionality requirements

(4) Design sketch

(5) Notes & open questions

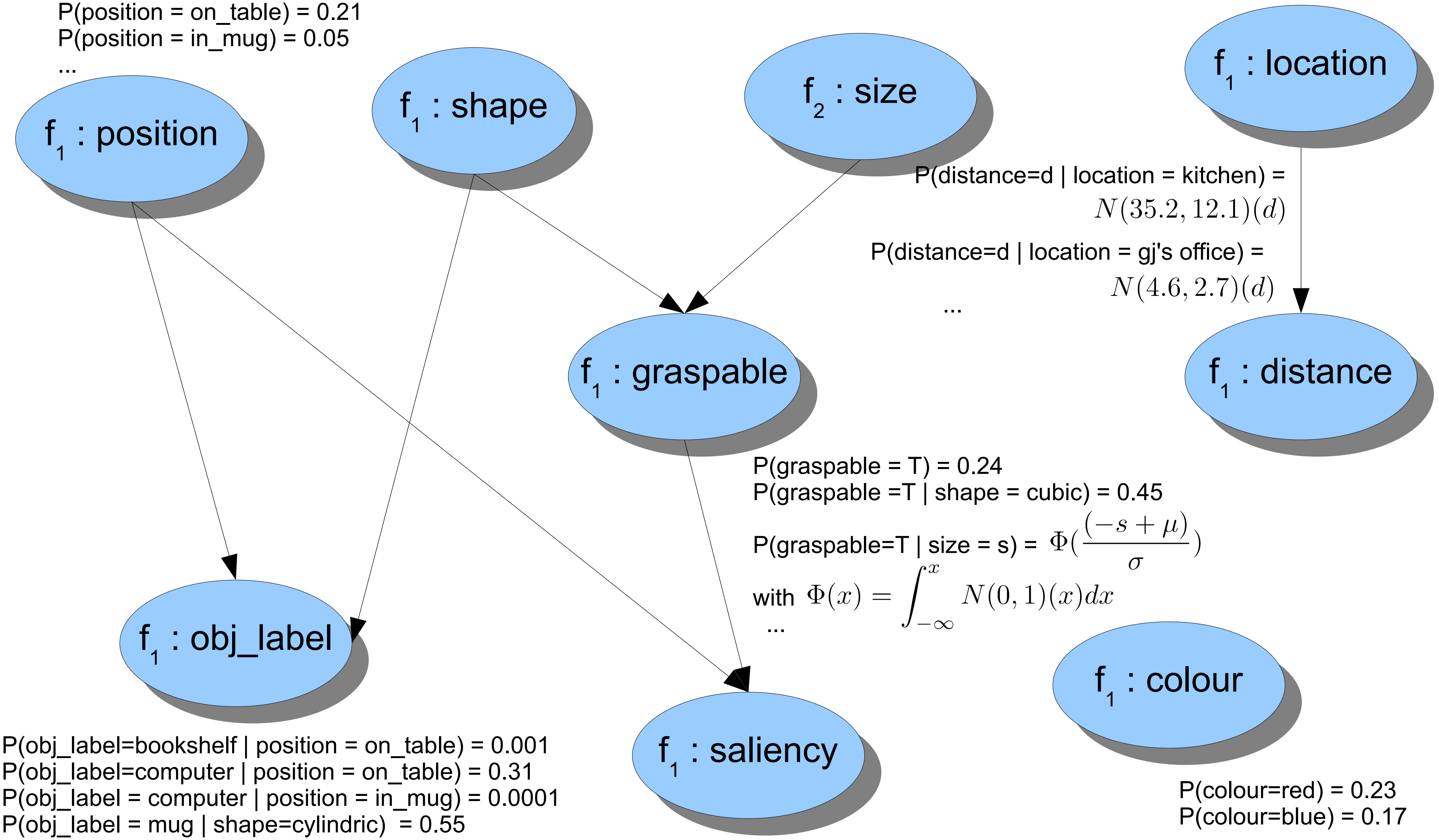
Estimating the prior probabilities

- As we have already seen, proxies and unions are both basically defined as lists of feature instances
- So the prior probability $P(\mathbf{f_{U_k}})$ is the joint probability of the features listed in the union:

$$P(\mathbf{f_{U_k}}) = P(f_{U_k}^1, f_{U_k}^2, ...f_{U_k}^o)$$
$$= \prod_{i=1}^{o} P(f_{U_k}^i | f_{U_k}^1 ... f_{U_k}^{i-1})$$

- These probabilities will be computed based on a multimodal *directed graphical model*, i.e. a **Bayesian network** of feature probabilities
- The role of this Bayesian network is to specify the *dependencies/correlations* between the features instances
  - For instance, to specify that the feature "*shape:cylindric*" is positively correlated to the feature "*obj_label=mug*"
  - Correlations both for *discrete* and *continuous* variables

A Bayesian network of features



P(position = on_table) = 0.21
P(position = in_mug) = 0.05
...

P(distance=d | location = kitchen) = $N(35.2, 12.1)(d)$
P(distance=d | location = gj's office) = $N(4.6, 2.7)(d)$
...

P(graspable = T) = 0.24
P(graspable =T | shape = cubic) = 0.45
P(graspable=T | size = s) = $\Phi(\frac{(-s+\mu)}{\sigma})$
with $\Phi(x) = \int_{-\infty}^{x} N(0,1)(x)dx$
...

P(obj_label=bookshelf | position = on_table) = 0.001
P(obj_label=computer | position = on_table) = 0.31
P(obj_label = computer | position = in_mug) = 0.0001
P(obj_label = mug | shape=cylindric) = 0.55
P(obj_label = mug | shape = cylindric, position = on_table) = 0.71
...

P(colour=red) = 0.23
P(colour=blue) = 0.17
....

Graphical models?

- Graphical models allow us to easily model the various kinds of interactions between modal features
- They offer a strong theoretical foundation for our binding system
- And, *last but not least*, there already exist a wide variety of efficient ML algorithms to learn graphical models (both concerning the *parameters* and the *structure* of the model)

“Graphical models are a marriage between probability theory and graph theory. They provide a natural tool for dealing with two problems that occur throughout applied mathematics and engineering -- **uncertainty** and **complexity** -- and in particular they are playing an increasingly important role in the design and analysis of machine learning algorithms. Fundamental to the idea of a graphical model is the notion of modularity -- a complex system is built by combining simpler parts. Probability theory provides the glue whereby the parts are combined, ensuring that the system as a whole is consistent, and providing ways to interface models to data. The graph theoretic side of graphical models provides both an intuitively appealing interface by which humans can model highly-interacting sets of variables as well as a data structure that lends itself naturally to the design of efficient general-purpose algorithms.”
Michael Jordan, “*Learning in Graphical Models*”, MIT Press, 1998.

Estimating the prior probabilities (2)

- The graphical model specifies the possible conditional dependencies between features values (across all possible modalities)
- Based on this information, computing the joint probability

$$\prod_{i=1}^{o} P(f^{i}_{U_k} | f^{1}_{U_k} \dots f^{i-1}_{U_k})$$

  becomes straightforward, and this gives us the value of the prior probability $\boxed{P(\mathbf{f_{U_k}})}$.
- This prior probability can be seen as a measure of the "*internal consistency*" of the union
  - An union with several highly correlated features will receive a much higher probability than one with conflicting/incompatible feature values.

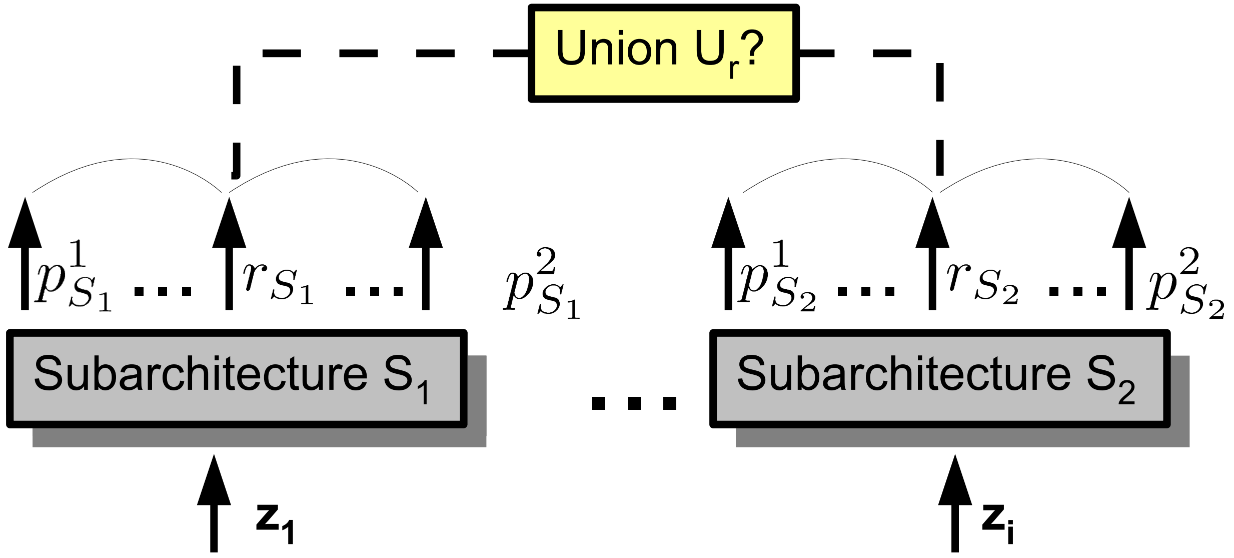
Dealing with relation proxies

- As we have seen, relation proxies have the particular property of having two features *source_proxy* and *target_proxy*
- The domain of these two features is the set of proxies generated by the subarchitecture from which the relation proxy originates
- Two relation proxies should be merged into a single (relation) union **iff**
  1. Their respective source proxies are bound into the same union
  2. Their respective target proxies are bound into the same union
  3. And provided the other features of the relation proxies are compatible as well
- But how can we enforce this constraint using the Bayesian network framework we presented?
  - We need to make sure that the prior probability P(U) of an union of two relation proxies is high if the source- and target-proxies of these relations are also bound in unions, and low otherwise

Dealing with relation proxies (2)

- Let $p^1_{S_1}$, $p^2_{S_1}$, $p^1_{S_2}$ and $p^1_{S_2}$ be four usual proxies, generated by S_1 and S_2, and let r_{S_1} and r_{S_2} be two relation proxies, with
  - $source\_proxy(r_{S_1}) = p^1_{S_1}$
  - $target\_proxy(r_{S_1}) = p^2_{S_1}$
  - $source\_proxy(r_{S_2}) = p^1_{S_2}$
  - $target\_proxy(r_{S_2}) = p^2_{S_2}$

- We want to compute the probability of an union U_r merging the two relation proxies r_{S_2} and r_{S_1} .



Dealing with relation proxies (3)

- This calculation depends on the prior probability $P(U_r)$, which includes the correlations between the features of both relation proxies (and hence also the correlations between the *source_proxy* and *target_proxy* features).
- To enforce the constraints expressed on the previous slide, we can therefore tune the two following conditional probabilities in the Bayesian network:

$$P(source\_proxy(r_{S_1}) = p^1_{S_1} \mid source\_proxy(r_{S_2}) = p^1_{S_2})$$
$$P(target\_proxy(r_{S_1}) = p^2_{S_1} \mid target\_proxy(r_{S_2}) = p^2_{S_2})$$

- The first (resp. second) probability will be set to a high level if it is the case that $p^1_{S_2}$ and $p^2_{S_2}$ (resp. $p^1_{S_1}$ and $p^2_{S_1}$) are bound into a single union U_1 (resp. U_2), and low otherwise
  - The requires an online *adaptation* of the Bayesian network.
  - Practically, the two conditional probabilities can be expressed as a function of the probabilities $P(U_1)$ and $P(U_2)$

Additional notes

- We can have dependencies between more than two features
- Since the graphical model will eventually contain both discrete and continuous variables, it will be formally defined as a **hybrid Bayesian network**

- **Q3**: In a graphical model, the sequential order in which the variables are placed is important. How can we set up the network such that the sequential order of the dependencies is optimal?

- **Q4**: (if such proxies are needed at all) How can we efficiently deal with *group proxies / group unions* ?

Outline

(1) Why a new design?

(2) A Bayesian approach to binding

- Proxies and unions
- Framing the binding problem
- How to estimate the parameters
  - Prior union probability
  - **Proxy probabilities**
- The Binding Algorithm
- Probabilistic reasoning over time

(3) Functionality requirements

(4) Design sketch

(5) Notes & open questions

Estimating the proxy probabilities

- The binding formula also requires us to provide the probability

$$\boxed{P(\mathbf{f}_{\mathbf{p}^{\mathbf{j}}_{\mathbf{S_i}}} \mid \mathbf{z_i})}$$

  for each proxy included in the union.

- This probability can be directly accessed via the information contained in the proxy:

Proxy $p^j_{S_i}$ **:**

$P(\text{Exists(a)=T}|\ \mathbf{z_i}) = 0.91$

$P(Shape(a)\text{=cylindric} \mid \text{Exists(a)=T},\ \mathbf{z_i}) = 0.87$

$P(Shape(a)\text{=spherical} \mid \text{Exists(a)=T},\ \mathbf{z_i}) = 0.11$

$P(Colour(a)\text{=red} \mid \text{Exists(a)=T},\ \mathbf{z_i}) = 0.94$

$P(Colour(a)\text{=blue} \mid \text{Exists(a)=T},\ \mathbf{z_i}) = 0.02$

$$P(\mathbf{f}_{\mathbf{p}^{\mathbf{j}}_{\mathbf{S_i}}} | \mathbf{z_i}) = \approx \left[\prod_{k=1}^{o} P(f^k_{p^j_{S_i}} \mid \text{Exists(a)}, \mathbf{z}) \right] \times P(\text{Exists(a)}\ |\mathbf{z_i})$$

Putting things together

- Finally, the prior probability of the proxy $\boxed{P(\mathbf{f}_{\mathbf{p}^{\mathbf{j}}_{\mathbf{S_i}}})}$ can be obtained via the graphical model, as for the unions.

- Using the probabilities $\boxed{P(\mathbf{f}_{\mathbf{U_k}})}$, $\boxed{P(\mathbf{f}_{\mathbf{p}^{\mathbf{j}}_{\mathbf{S_i}}} \mid \mathbf{z_i})}$ and $\boxed{P(\mathbf{f}_{\mathbf{p}^{\mathbf{j}}_{\mathbf{S_i}}})}$, we can then finally compute the probability of the union given all the accumulated evidences:

$$P(\mathbf{f}_{\mathbf{U_k}} \mid \mathbf{z_1}, ...\mathbf{z_n}) = \beta\ P(\mathbf{f}_{\mathbf{U_k}}) \times \prod_{\{p^j_{S_i} \in \text{ proxies}(U_k)\}} \frac{P(\mathbf{f}_{\mathbf{p}^{\mathbf{j}}_{\mathbf{S_i}}} \mid \mathbf{z_i})}{P(\mathbf{f}_{\mathbf{p}^{\mathbf{j}}_{\mathbf{S_i}}})}$$

(the normalisation constant can be left out)

Outline

(1) Why a new design?

(2) A Bayesian approach to binding

- Proxies and unions
- Framing the binding problem
- How to estimate the parameters
- **The Binding Algorithm**
- Probabilistic reasoning over time

(3) Functionality requirements

(4) Design sketch

(5) Notes & open questions

Moving forward

- The formula we derived allow us to compute the probability of a specific union.
- But we still have to face two problems:

1. First, the formula only gives us a multivariate probability distribution, not a specific probability value. If we want to compare the likelihood of different unions, we need to compute the *maximum value* (highest peak in a high-dimensional space) taken by the probability distribution.

   → **Q5**: What is the optimal method to achieve this? Gradient descent on the probability distribution? Or are there more efficient methods of doing so ? For discrete features, the optimization can be easily implemented (via enumeration) – but as soon as we include continuous features...

2. Second, we don't yet know which unions we need to "test". The search space of potential unions can be quite large, so we have to strongly constrain our search! i.e., we need a *scalable* and *efficient* algorithm to produce possible unions for which we can then compute the probability.

   → See next slide

The Binding algorithm

Algorithm 1 Binding Algorithm

```
Require: - A list of subarchitectures $S_1 ... S_n$
         - For each subarchitecture $S_i$, a set of proxies $p^1_{S_i} ... p^m_{S_i}$
         - $getmax()$ is a function returning the maximum value taken by a
           multivariate probability distribution
         - $curBindings$ is a hashmap returning, for each proxy, the union
           including it with the current maximal probability.

  % curBindings is initialized with the union containing only this unique proxy
  for $p^j_{S_i} \in proxies(S_1 ... S_n)$ do
    $curBindings(p^j_{S_i}) \leftarrow getmax(P(p^j_{S_i} \mid \mathbf{z_i}))$
  end for

  for $i_1 = 1 ... (n-1)$ do
    for $p_1 \in proxies(S_{i_1})$ do
      for $i_2 = (i_1 + 1) ... n$ do
        for $p_2 \in proxies(S_{i_2})$ do

          Compute $P(U \mid \mathbf{z_1}, \mathbf{z_2}) = getmax\left(P(U) \frac{P(p_1 \mid \mathbf{z_1}) \; P(p_2 \mid \mathbf{z_2})}{P(p_1) \; P(p_2)}\right)$

          if $P(U \mid \mathbf{z_1}, \mathbf{z_2}) > [curBindings(p_1) \times curBindings(p_2)]$ then
            $curBindings(p_1) \leftarrow P(U \mid \mathbf{z_1}, \mathbf{z_2})$
            $curBindings(p_1) \leftarrow P(U \mid \mathbf{z_1}, \mathbf{z_2})$
          end if

        end for
      end for
    end for
  end for

  % (This process can be repeated on the resulting unions to get higher-order unions)
  return  $curBindings(proxies(S_1 ... S_n))$
```

Outline

(1) Why a new design?

(2) A Bayesian approach to binding

- Proxies and unions
- Framing the binding problem
- How to estimate the parameters
- The Binding Algorithm
- **Probabilistic reasoning over time**

(3) Functionality requirements

(4) Design sketch

(5) Notes & open questions

Probabilistic reasoning over time

- The content of the binder WM will change over time, reflecting the changes in the environment, and how the dialogue unfolds
- But we know that the proxies generated by some subarchitectures might be temporally unstable
- Question: how can we gradually accommodate these changes while maintaining a certain stability in the binder WM content?
- This would require the use of probabilistic temporal models, as well as algorithms for efficient Bayesian filtering
- Use *dynamic Bayesian networks* to reason under uncertainty over time?

→ Still have to complete this section!

Outline

(1) Why a new design?

(2) A Bayesian approach to binding

(3) Functionality requirements

(4) Design sketch

(5) Notes & open questions

Functionality requirements

- The next slides detail a list of requirements for the binder subarchitecture
- These are split in five lists:
  - Update of binder content
  - Inspection / extraction of binder content
  - Binding mechanisms
  - Parametrization of binder model
  - Usability aspects
- For each requirement, we specify the software release in which it is expected to be included:
  - release 1 : mid-July
  - release 2 : September
  - release 3 : After the review meeting

Update of binder content

- Mechanisms for the insertion of modal proxies (structured as list of features with probabilities) by the subarchitectures

release 1	1. Intuitive format for specifying the multivariate probability distribution of the proxy (Ice data structure + utility libraries)
release 1	2. Inclusion of discrete features
release 2	3. Inclusion of continuous features, based on a set of usual parametrized probabilistic distributions (Gaussian, etc.)
release 2	4. Dedicated data structures and libraries for modelling saliency and spatio-temporal frames
release 2	5. Use of ontologies (e.g. from coma.sa) in the assignment of feature values
release 2	6. Insertion of relation proxies
release 3	7. Filtering mechanisms to prune unlikely events from the proxy distribution, and hence improve the binding efficiency
release 3	8. Incremental extension / refinement of an existing proxy via the insertion of new features into the proxy (using dynamic programming techniques to avoid having to completely recompute the binds)
release 3	9. If needed, insertion of group proxies
release 3	10. Mechanisms for proxy (re)identification (“tracking”) over time

Inspection of binder content

- Mechanisms for efficient inspection / extraction of binder content by the subarchitectures

release 1

1. Common methods for accessing the WM content (easy retrieval of the union data structures, calculation of the maximum probability value of an union)

release 1

2. Module for translating the content of the binder WM onto a *discrete symbolic state* (to be used e.g. by the continual planner, or to perform context-sensitive processing in the subarchitectures). This require the possibility to “collapse” the probability distributions over unions into discrete units.

release 2

3. Automatic detection of possible binding *ambiguities* based on the probability distributions of unions.

release 2

4. Specification of a binding *query language*, and integration of a module taking these queries as input

release 2

5. “What-if” predictions: evaluating the probability of particular unions if a new proxy was to appear (but without actually modifying anything in the WM)

release 3

6. Module for storage and retrieval of binding history over extended periods of time, with detailed logging information on the perceived spatio-temporal frames (to be used e.g. for offline learning and adaptation)

release 3

7. Automatic detection of knowledge gaps in binding WM?

Binding mechanisms

- Mechanisms for cross-modal information binding

1. Module devoted to the efficient calculation of the prior probability of the union, based on the graphical model (exact inference, approximation via Markov Chain Monte Carlo or Gibbs sampling?)
    - release 1 — Limited to discrete sets of features
    - release 2 — Extended to mixed discrete and continuous features
2. release 1 — Implementation of the binding formula (p. 36), based on the Independent Likelihood Pool technique
3. release 1 — Module devoted to the calculation of the maximum value taken by a multivariate mixed probability distribution (gradient descent?), in order to extract the maximum probability of an union
4. release 1 — Implementation of **Algorithm 1** (p. 55), which enumerates the list of possible unions and evaluates them one-by-one
5. release 2 — Filtering mechanisms to prune unlikely unions (based on obvious incompatibility patterns) without explicit probabilistic calculations
6. release 3 — Dealing with temporal instabilities via probabilistic reasoning over time (dynamic Bayesian network?)
7. release 3 — Lazy binding: “on-demand” update of binder content, based on top-down attentional state combined with the usual bottom-up perceptive processes
8. release 3 — Cross-modal (re)identification and tracking of unions over extended periods of time

Parametrisation of binder model

- Mechanisms to easily parametrise / adapt / learn the probabilistic model included in the binder:

release 1
1. Simple specification format for the graphical model

release 1
2. Visualization GUI of the graphical model

   (based on the open-source Java library JGraph?)

release 2
3. Inclusion in the graphical model of *continuous features*

release 2
4. Inclusion in the graphical model of features for *relations proxies*

release 3
5. If needed, inclusion in the graphical model of features for *group proxies*

6. Machine learning algorithms to automatically *learn* the cross-modal graphical model:

   release 3
   1. Learning the *parameters* of the model with a set of offline training examples

   release 3
   2. Learning the *parameters* of the model with offline examples + online socially-guided learning

   release 3
   3. Extension of the approach to learn both the *parameters* and the *structure* of the model

release 3
7. Continuous online *adaptation* of the model

Usability aspects

- Usability requirements of the software to be developed:

release 1 — 1. Clean, extensible design, centered around a small set of core components

release 1 — 2. Fully documented API describing the interactions between the binder and the subarchitectures (update and inspection of binder content)

release 1 — 3. Fully documented data structures specified in Ice, using the inheritance capabilities provided in the language to structure the representations

4. GUI visualization and control module (using Swing and Jgraph?)
    - release 1 — Visualisation window to see how the unions are being created, and how the WM evolves over time
    - release 2 — Possibility to insert / modify proxies on the fly

5. Binding must be made as *fast* and *efficient* as possible
    - release 1 — Careful design of the main algorithms to ensure optimal behaviour
    - release 2 — Possibility to set various parameters at runtime to control the binding behaviour (pruning and filtering mechanisms)

release 1 — 6. Fake monitors to test the binder behaviour

release 2 — 7. Intensive black-box test suite

release 1 — 8. Javadoc-compatible code documentation

Outline

(1) Why a new design?

(2) A Bayesian approach to binding

(3) Functionality requirements

(4) Design sketch

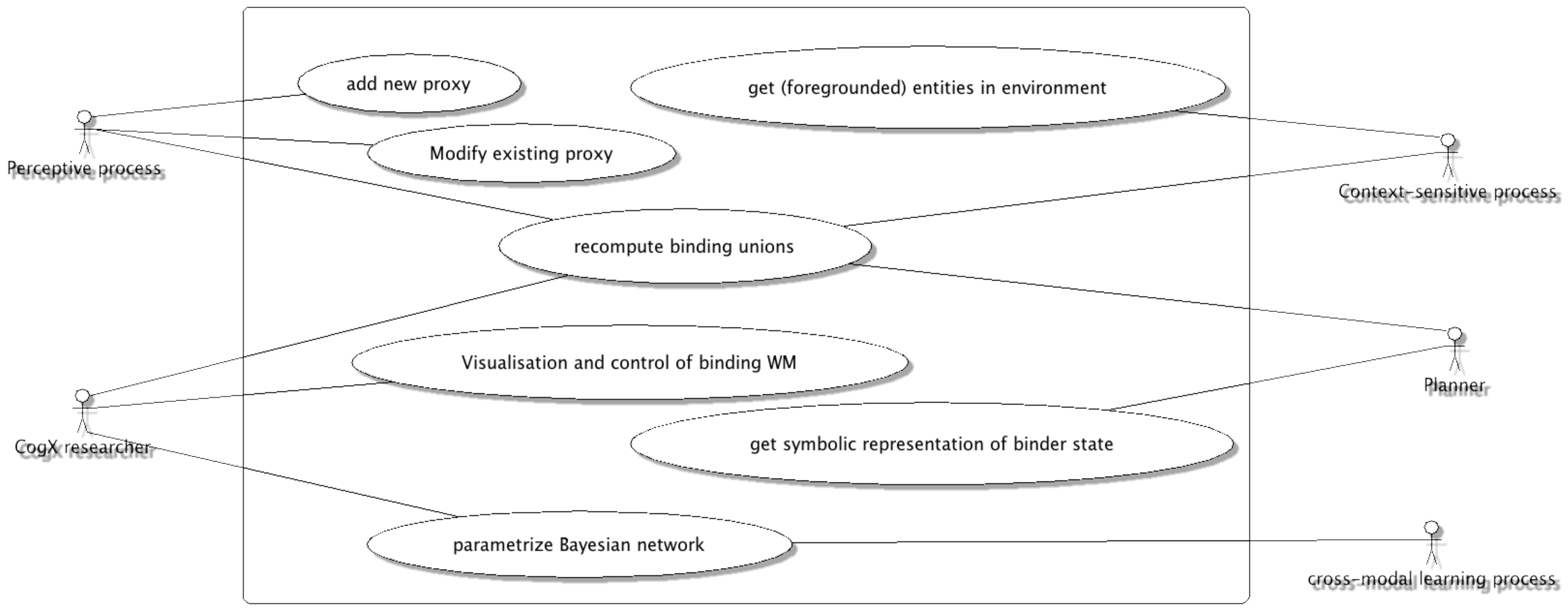
(5) Notes & open questions

Requirements engineering

- The next slides detail a first design sketch (via UML diagrams) of the binder subarchitecture
- The current design is limited to release 1 (mid-July), but leaves enough room for the gradual extensions and refinements planned for the releases 2 and 3
- This section should be enriched (UML state-chart diagrams, details on the data structures, and most importantly, a first API)

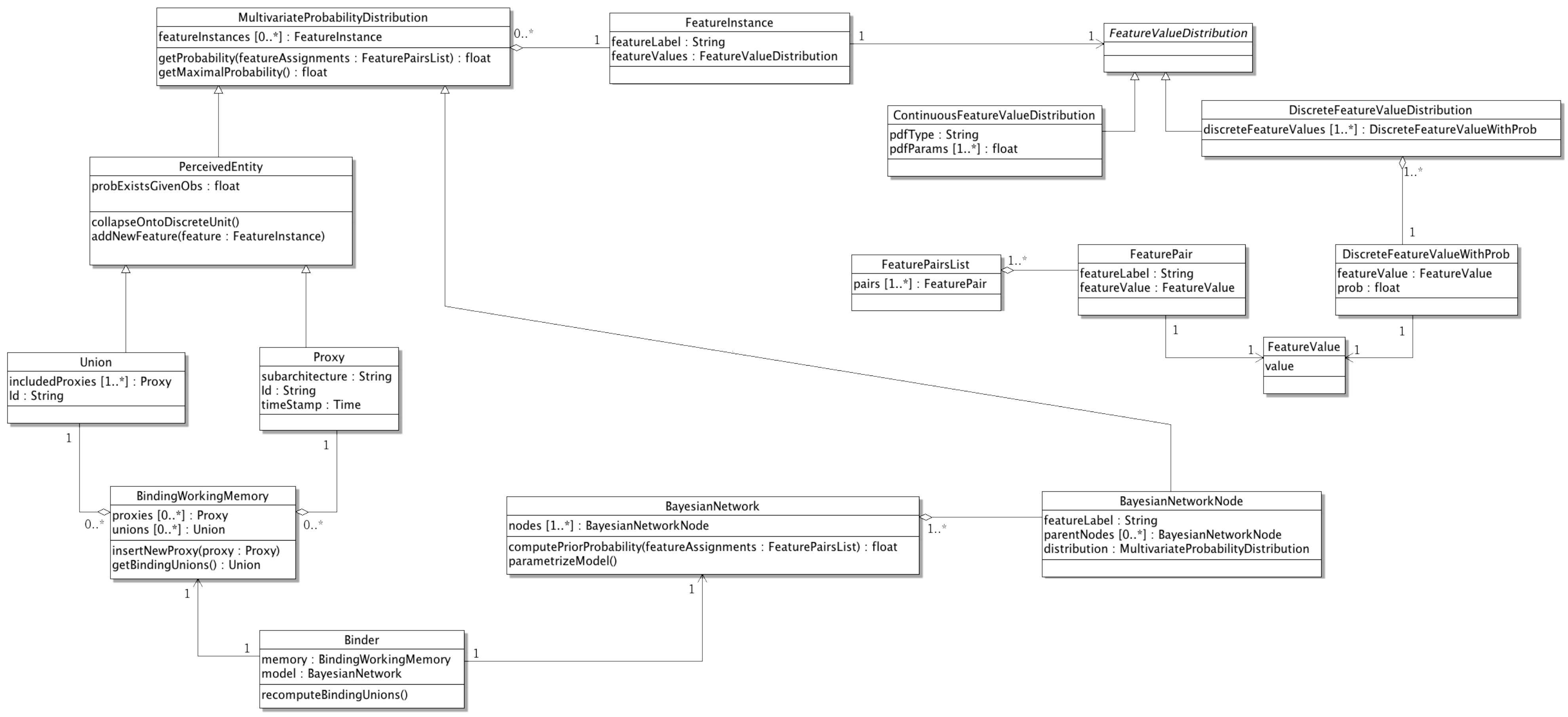
Use case diagram

- A simple UML use case diagram for the binder


add new proxy
get (foregrounded) entities in environment
Perceptive process
Modify existing proxy
Context-sensitive process
recompute binding unions
Visualisation and control of binding WM
Planner
CogX researcher
get symbolic representation of binder state
parametrize Bayesian network
cross-modal learning process

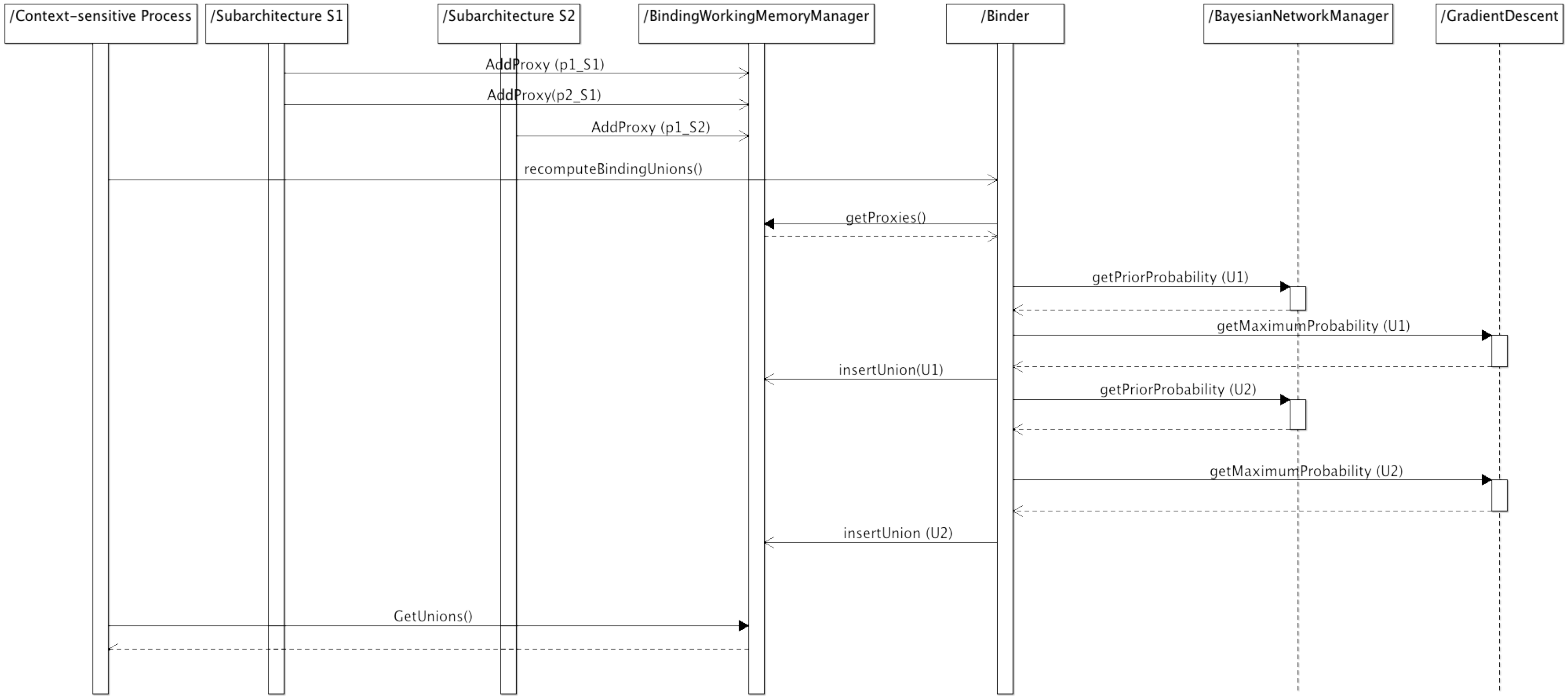
Class diagram

- A (slightly less simple) UML class diagram:


MultivariateProbabilityDistribution
featureInstances [0..*] : FeatureInstance
getProbability(featureAssignments : FeaturePairsList) : float
getMaximalProbability() : float
0..*
1
FeatureInstance
featureLabel : String
featureValues : FeatureValueDistribution
1
1
FeatureValueDistribution
ContinuousFeatureValueDistribution
pdfType : String
pdfParams [1..*] : float
DiscreteFeatureValueDistribution
discreteFeatureValues [1..*] : DiscreteFeatureValueWithProb
1..*
1
PerceivedEntity
probExistsGivenObs : float
collapseOntoDiscreteUnit()
addNewFeature(feature : FeatureInstance)
FeaturePairsList
pairs [1..*] : FeaturePair
1..*
FeaturePair
featureLabel : String
featureValue : FeatureValue
DiscreteFeatureValueWithProb
featureValue : FeatureValue
prob : float
1
1
1
1
FeatureValue
value
Union
includedProxies [1..*] : Proxy
Id : String
Proxy
subarchitecture : String
Id : String
timeStamp : Time
1
1
BindingWorkingMemory
proxies [0..*] : Proxy
unions [0..*] : Union
insertNewProxy(proxy : Proxy)
getBindingUnions() : Union
0..*
0..*
BayesianNetwork
nodes [1..*] : BayesianNetworkNode
computePriorProbability(featureAssignments : FeaturePairsList) : float
parametrizeModel()
1..*
BayesianNetworkNode
featureLabel : String
parentNodes [0..*] : BayesianNetworkNode
distribution : MultivariateProbabilityDistribution
1
1
Binder
memory : BindingWorkingMemory
model : BayesianNetwork
recomputeBindingUnions()
1
1

Sequence diagram

- UML sequence diagram



Outline

(1) Why a new design?

(2) A Bayesian approach to binding

(3) Functionality requirements

(4) Design sketch

(5) Notes & open questions

Modelling saliency

→ To be written

To do:

- Explain how to model saliency at a modality-specific level (what kind of continuous probabilistic distribution to use, with which parameters)
- Explain how these saliencies are merged in the binder working memory to yield a cross-modal saliency, which could take the form of a multivariate continuous distribution
- Explain how this saliency information can be fruitfully exploited by the subarchitectures to implement context-sensitive processing

Modelling spatio-temporal frames

→ To be written

To do:

- Explain how we can “cast” the binding approach within gj's formalization of a *belief model*
- Also investigate the possible interactions between the binding WM and the *episode-like, long-term memory*

Binding over time

- The binding approach we outlined is mainly concerned with the binding of *different* modalities *at the same point in time*. Another kind of "binding" we haven't really touched is how to bind information (from one or several modalities) over *extended periods of time*.
  - For instance, detecting that the object we are currently perceiving is the same as the one we perceived 10 minutes ago
  - This would probably require the inclusion of a sophisticated word model (the tracking of a moving object would for instance require some modelling of the laws of physics)
  - Note that this problem is conceptually different from the one outlined in the section "Probabilistic reasoning over time".
  - The dynamic Bayesian network can indeed only be helpful for very brief temporary instabilities, but cannot perform any real *tracking* of objects over extended periods of time.
  - **Q6**: How could we efficiently implement such form of binding ?

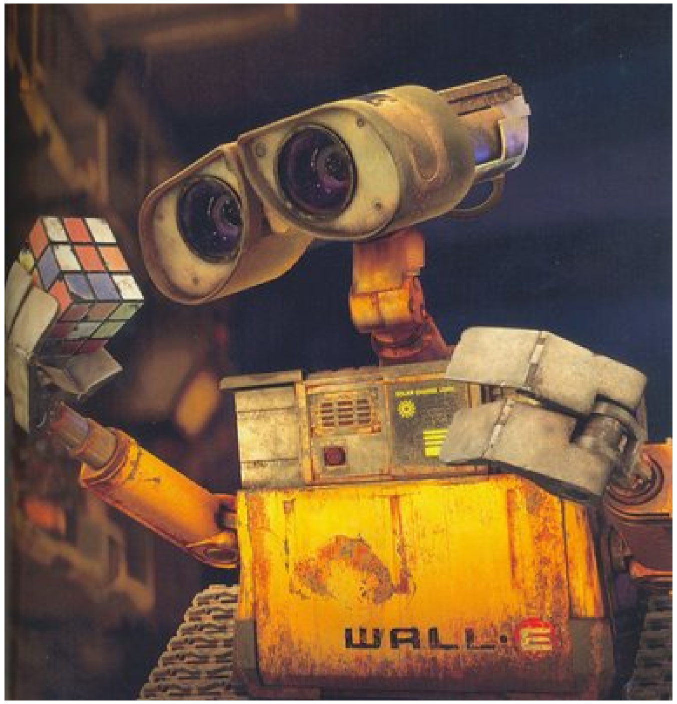
Semiotics

- On a more theoretical level, it would be interesting to see how we can define binding as a more general (Peircian) *semiotic process*, and see what this exactly entails
  - What we are effectively doing in binding is indeed a *two-step interpretation process*: one arising at the subarchitecture level (low-level features interpreted as set of proxy features, given the modal word model), and then the set of modal interpretations is reinterpreted in the binder to yield a cross-model interpretation.
- A more technical question is related to the use of rich ontologies in specifying some feature values
  - The feature “obj_label” could for instance receive values extracted from an ontology of possible physical entities (expressing the fact that a mug is a specific type of container, which is itself a subtype of an “object”, that it is also a graspable object, etc.).
  - **Q7**: how can we effectively use such ontologies in our probabilistic model?

Cross-modal learning

- And of course, we haven't talked here about the crucial *learning* aspects – at first, we can manually specify the feature correlations in the bayesian network, but in the long term, the correlations should be learned automatically!
  - **Q8**: What kind of algorithms should be used for this type of cross-modal learning? Unsupervised, semi-supervised, reinforcement, socially-guided learning? A combination of these?
  - And on which data should we train the probabilistic model?

Happy end



That's all for now folks !

Please don't hesitate to send me your comments & suggestions, at pierre.lison@dfki.de !