Flexible Widget Layout with Fuzzy Constraint Satisfaction

Takuto YANAGIDA and Hidetoshi NONAKA
Hokkaido University, Japan

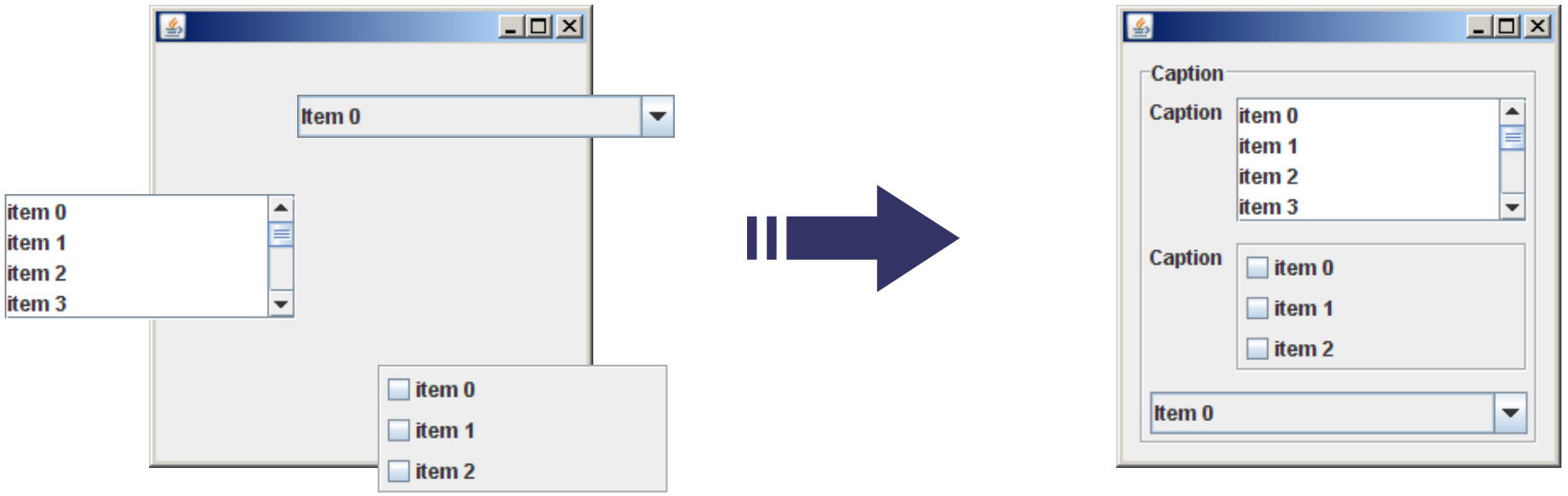
I. INTRODUCTION

Background

- Widget layout performed by computers is one of the most important challenges [1] for automatic generation of graphical user interfaces (GUIs).
- The layout has a significant impact on the usability of GUI applications and services, and it decides how easy to use them.

[1] S. Lok and S. Feiner, “A survey of automated layout techniques for information presentations,” in SmartGraphics ’01, 2001.

- Widget layout problem
  - is the process of deciding the positions and sizes of widgets, such as list boxes, radio buttons, and panels for grouping them.



Model-based user interface design

- In the field of model-based user interface (UI) design [2, 3], systems generate GUIs from logical descriptions, which do not specify which widgets to be used.
- Hence, selecting widgets is needed, and widget layout is more complicated.

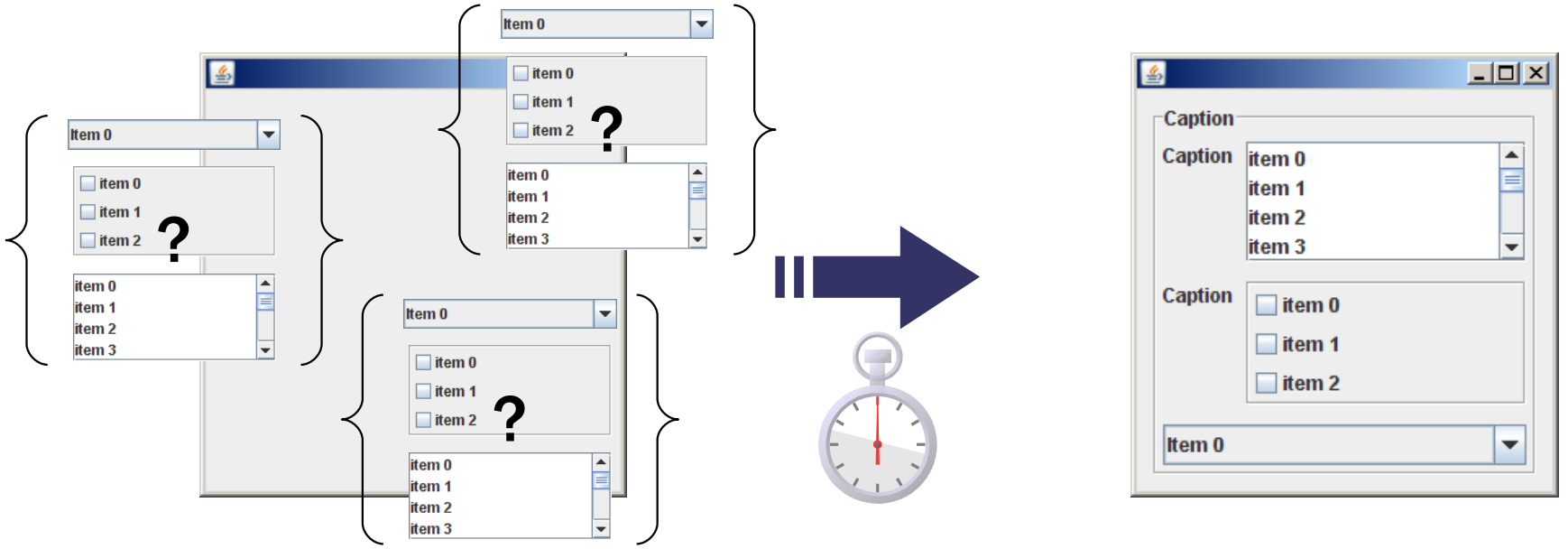
[2] J. Eisenstein, J. Vanderdonckt, and A. Puerta, “Applying model-based techniques to the development of UIs for mobile computers,” in IUI ’01, 2001.
[3] J. M. Vanderdonckt and F. Bodart, “Encapsulating knowledge for intelligent automatic interaction objects selection,” in CHI ’93, 1993.

Flexible widget layout

- Automatic GUI generation from logical descriptions requires both
  - deciding which widget and their alignments are used,
  - completing the layout in a certain time especially when the system generates them in run time.

Flexible widget layout (FWL)

- For FWL, a system searches combinations of widgets and their alignments selecting from their candidates.
- This feature enables a system to select small widgets with little usability for small screens, or large ones with enough usability for large screens.



Point of our proposal

- We formulate FWL problem as a **fuzzy constraint satisfaction problem** (FCSP) [12] in the field of artificial intelligence.
- We represent the desirability of the selections straightforward as **fuzzy constraints**; therefore, we can utilize existing techniques of FCSP.
- Our system generates GUI dialog boxes from *UI models* of logical descriptions.

[12] Z. Ruttkay, "Fuzzy constraint satisfaction," in Proceedings 1st IEEE Conference on Evolutionary Computing, Orlando, 1994, pp. 542–547.

II. FLEXIBLE WIDGET LAYOUT PROBLEM

FWL problem

- Appropriate widgets and their alignments are selected from sets of candidates.
- A set of widget candidates corresponds to a certain UI function, and every widget in the set represents the same function.
- FWL is executed based on a UI model or its descriptions, which contains UI functions and their groupings.

- The complexity of FWL is caused by that widgets with the trade-off between their desirability α and the ease of layout involving their dimensions.

easy > not easy

$$\alpha_{ddlb} < \alpha_{rbs}$$

User interface model

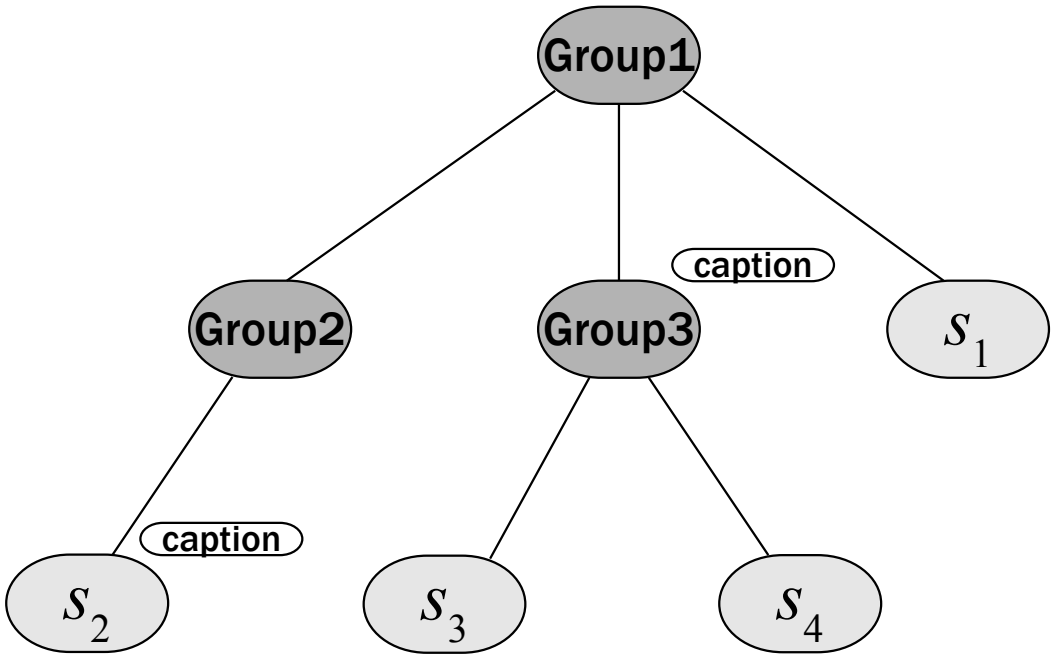
- As a UI model generally expressed in logical descriptions, in this paper, we adopt selection act model [5].
- In this model, UI functions are represented as selection acts with some parameters, and they are grouped to make a tree graph.

[5] T. Yanagida, H. Nonaka, and M. Kurihara, “User-preferred interface design with abstract interaction description language,” in IEEE International Conference on Systems, Man and Cybernetics, 2006.

- **Selection act s_i consists of:**
  - **list of choices L_i**
  - **number of selected items e_i**
  - **importance t_i**
  - **flag whether its choices have opposite meanings o_i**

$$s_i = \langle L_i, e_i, t_i, o_i \rangle$$

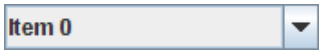
- All selection acts are grouped and make a tree graph of UI functions, whose root is a group, and it will correspond to a dialog box to be generated.
- Selection acts and the groups can have a caption string for their explanations.



Used widgets

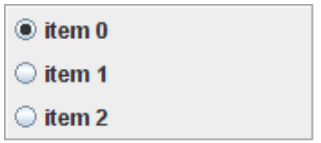
- Since they are commonly adopted by many existing toolkits, we use the subset of widgets.
  - We defined the desirability (usability) $0 \leq \alpha \leq 1$ corresponding to the types of widgets.

$$\alpha_{ddlb} < \alpha_{lb\_min} < \alpha_{lb\_max} < \alpha_{rbs} = \alpha_{cbs} < \alpha_{cb}$$

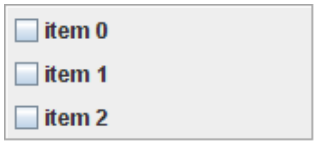


Drop down list box

List box



Radio buttons



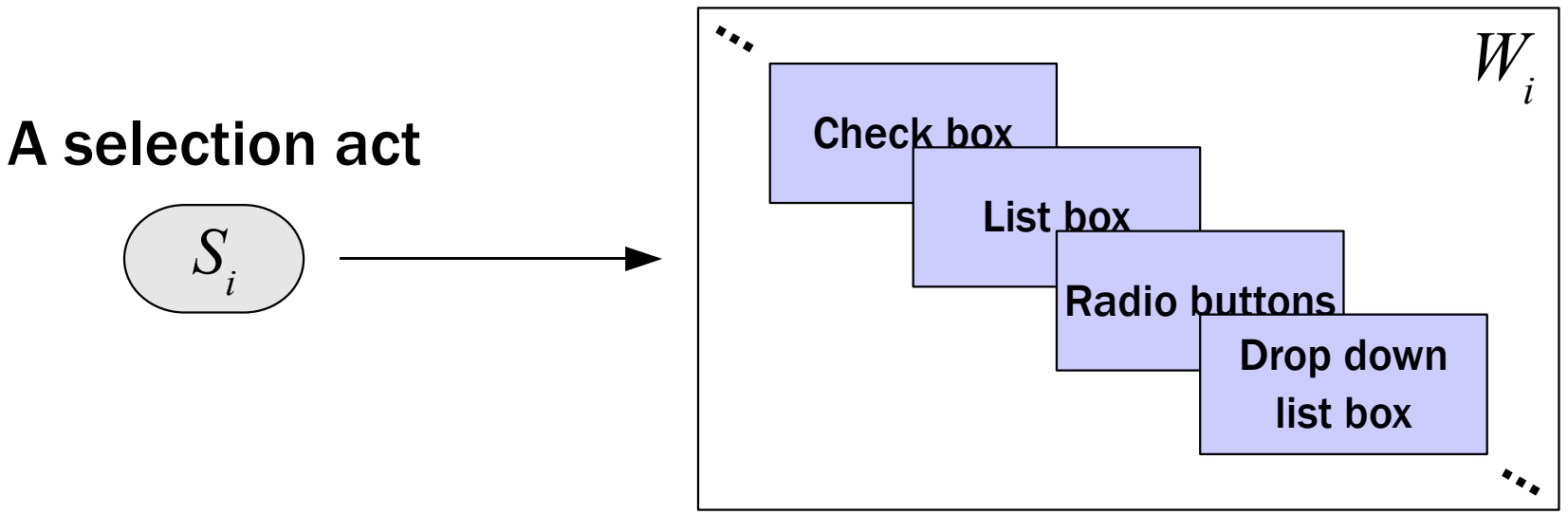
Check boxes

Check box

* there is a range of desirability for list box

Relation between model and widgets

- A Selection act is mapped to the corresponding set of widget candidates W_i, and it will be expressed with widget $w_i \in W_i$.



- **Widget candidates are chosen based on selection acts (TABLE 1).**

TABLE 1

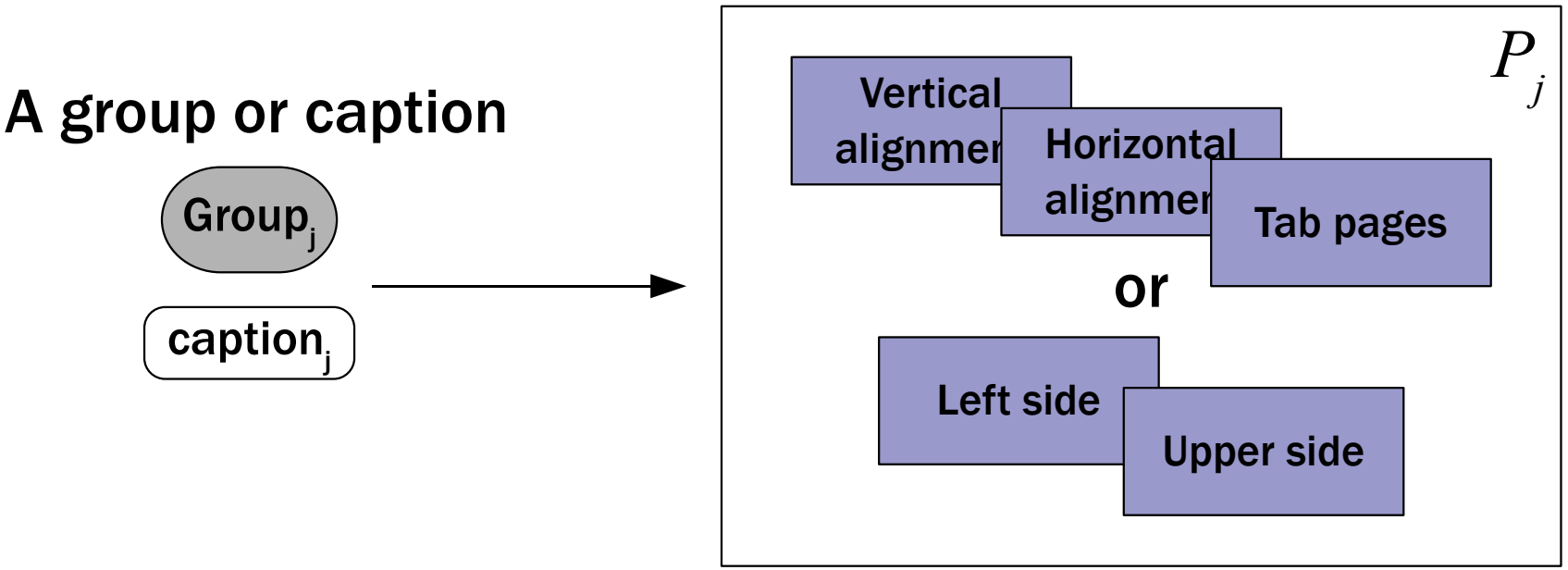
Selection size e_i	Item size $\|L_i\|$	Is opposite o_i	Candidates W_i
`single`	$\|L_i\| = 2$	`true`	Check box, Radio buttons, Drop down list box
		`false`	Radio buttons, Drop down list box
	$\|L_i\| < 6$	-	Radio buttons, Drop down list box
	$\|L_i\| \geq 6$	-	List box, Drop down list box
`multiple`	-	-	Check boxes, List box

- Each instance of widget w_i has a **minimum size** (width: w_{wi}, height: h_{wi}) uniquely defined by parameters of the corresponding selection act s_i.

Relation between model and positioning

- A group in UI models and captions are represented as array containers and labeled containers respectively.
- We express the caption of a selection act as a labeled container wrapping one element, because it also has positioning candidates.

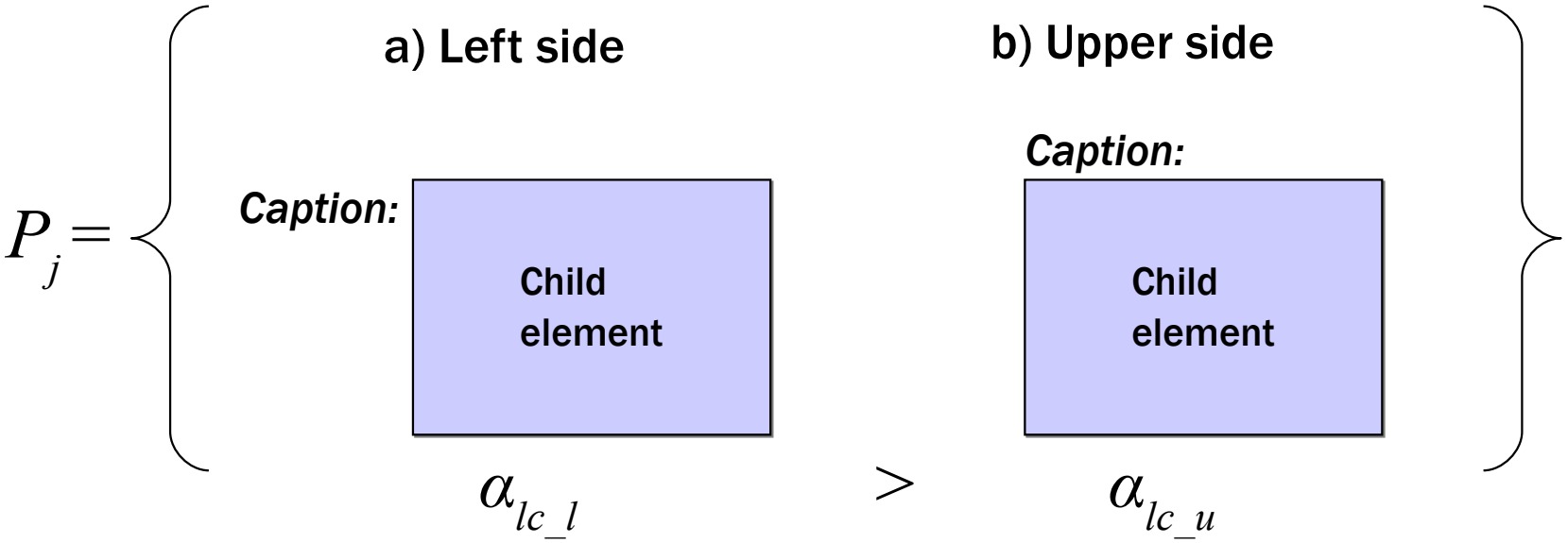
- A container is mapped to a set of positioning candidates P_j, and it will be expressed with positioning $p_j \in P_j$.



- An array container has three positioning candidates, and it aligns its child elements (widgets and containers).
- Desirabilities for the positioning candidates are defined.

$$P_j = \left\{ \begin{array}{ccccc} \text{a) Vertical alignment} & & \text{b) Horizontal alignment} & & \text{c) Tab pages} \\ \text{Child Element 1} & & \text{CE1} \; \text{CE2} \; \text{CE3} & & \text{CE1} \\ \text{CE2} & & & & \\ \text{CE3} & & & & \\ \alpha_{ac\_v} & = & \alpha_{ac\_h} & > & \alpha_{ac\_t} \end{array} \right.$$

- A Labeled container contains only one child element and has two candidates: the left side label and the upper side label.
- The desirabilities for the positioning candidates are also defined.



- Each positioning candidate p_j has
  - a minimum size (width: w_{pj}, height: h_{pj}) uniquely defined by the minimum sizes of its child elements and the length of caption if it is a labeled container.
  - maximum sizes for its children (width: $\mathrm{W}_{pj,\,1}$, ..., height: $\mathrm{H}_{pj,\,1}$, ...)

Possibility of doing layout

- The minimum sizes of widgets and containers decide whether it is possible to do a layout defined by selections from candidates.
- Solving FWL problem is finding the best combination of the candidates, which is *layout-possible* and has the *highest desirability*.

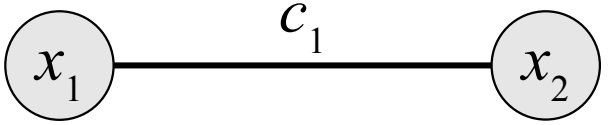
- Possibility of layout
  - means whether or not the child elements of a container can be placed in its rectangle when given a combination of candidates.

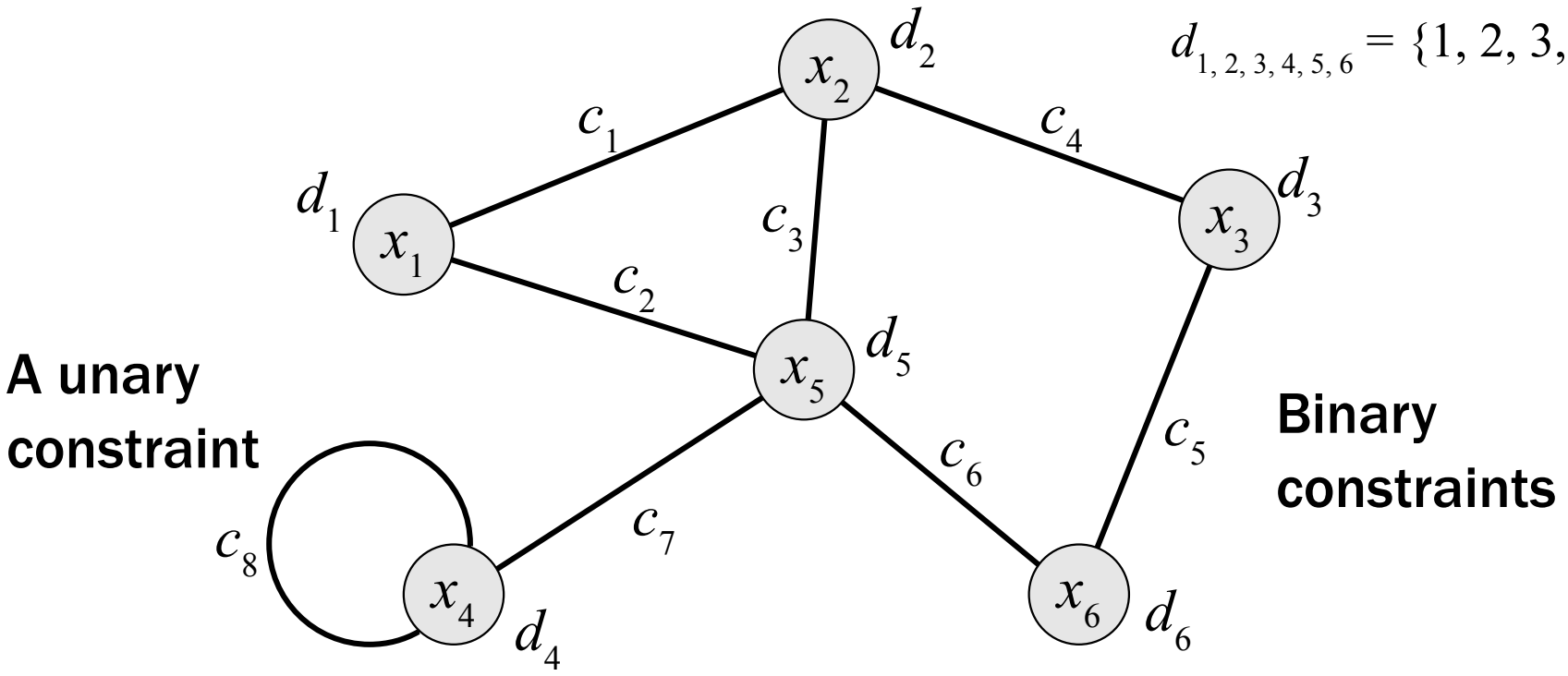
    (maximum size for $child_n$ ≥ minimum size of $child_n$)
- Desirability of layout
  - means how good usability the layout offers, and
  - is the minimum of desirabilities of selected candidates.

III. FORMULATION

A. Fuzzy constraint satisfaction

- **Fuzzy constraint satisfaction problem (FCSP)**
  - is a branch of combinatorial search problems
  - consists of
    - a set of variables $X = \{x_1, \dots, x_q\}$
    - a set of domains $D = \{d_1, \dots, d_q\}$
    - a set of constraints $C = \{c_1, \dots, c_r\}$
  - can be represented by a graph, where nodes and edges are corresponding to variables and constraints.



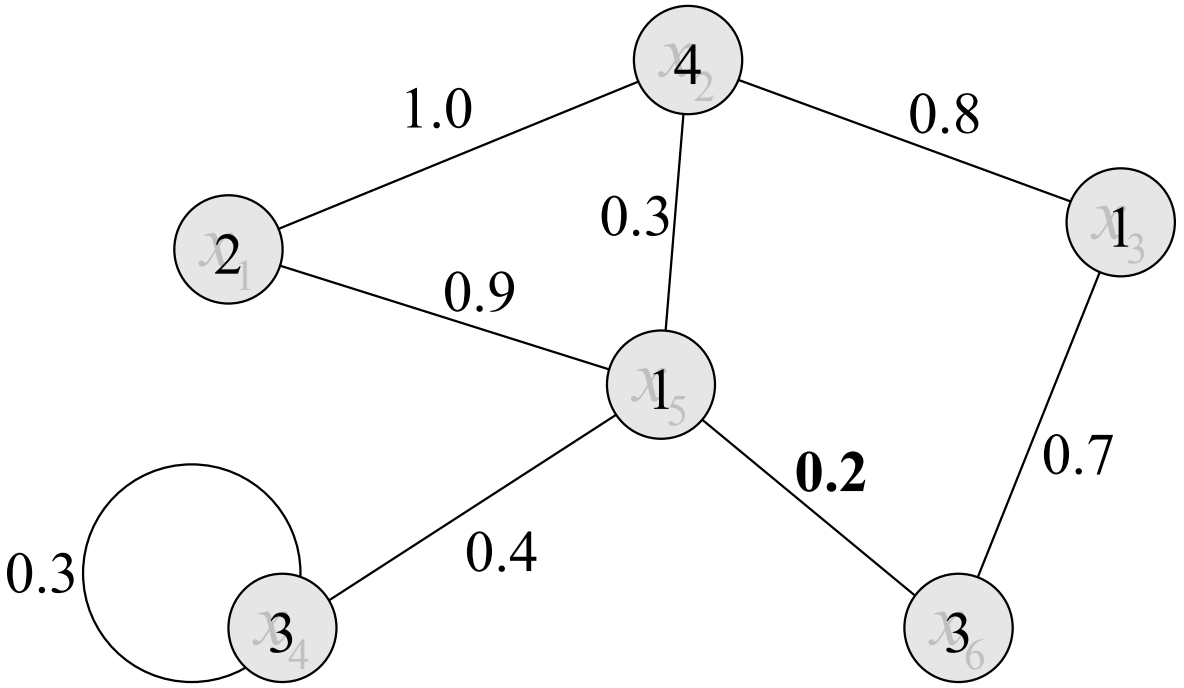


- c_h donates membership function $\mu R_h(v[S_h])$
  - S_h: scope (variables related to c_h)
  - v: assignment for all variables
  - A membership value is called a satisfaction degree.

- **A solution of a FCSP**
  - **The satisfaction degree of a whole FCSP is defined as a minimum of all constraint satisfaction degrees.**

$$Cmin(v) = \min(\mu R_h(v[S_h]))$$

  - **If $Cmin(v) > 0$, v is a solution of the FCSP.**



This assignment is a solution. Its satisfaction degree is 0.2.

B. Flexible widget layout with FCSP

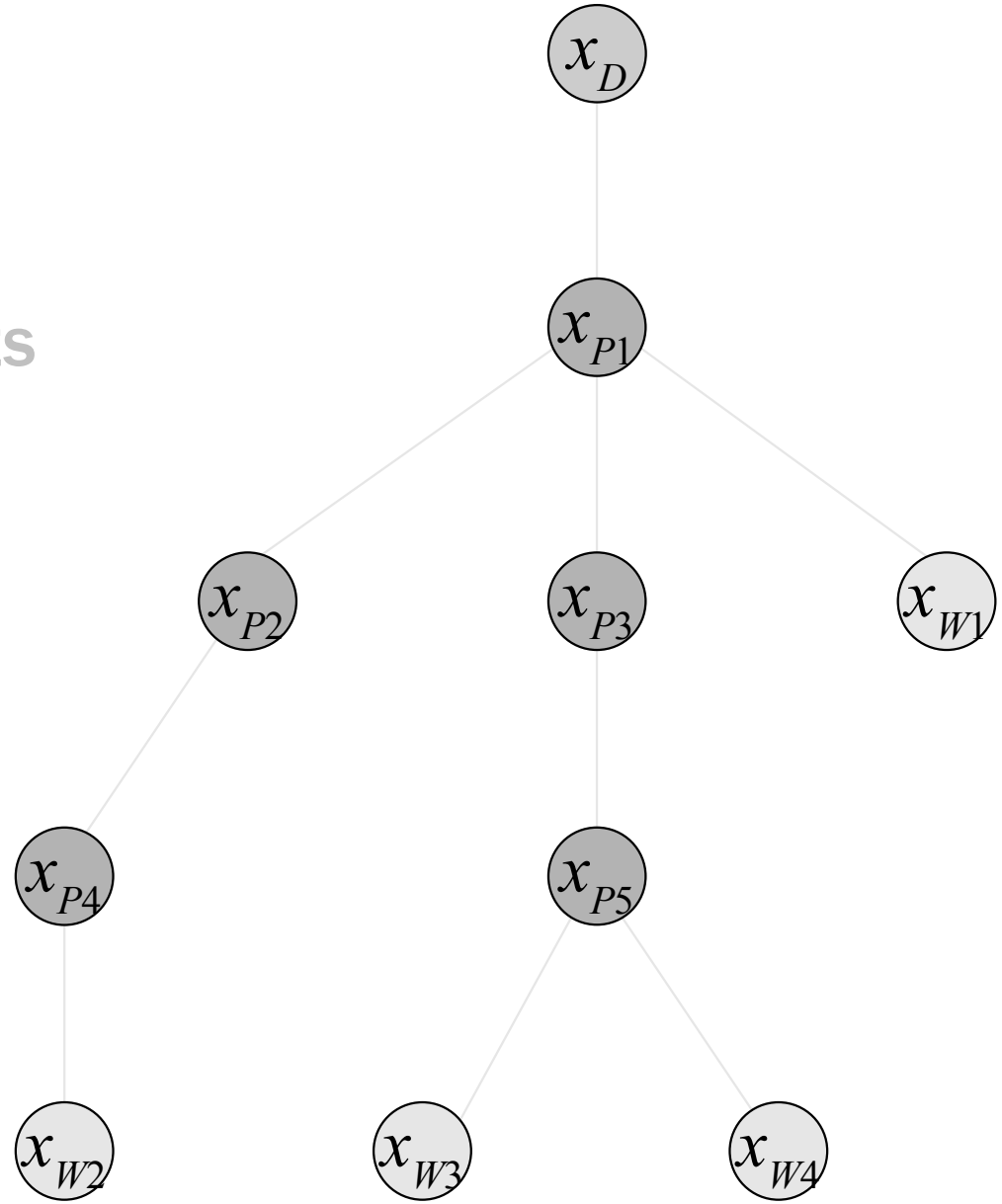
- We introduce the framework of FCSP, formulate FWL problems as FCSPs.
  - We use unary fuzzy constraints for expressing the desirability α of widgets.
  - We represent the parental relationship among widgets with binary crisp constraints, which are particular cases of fuzzy constraints.

Definition of variables

- **Variables $X = X_W \cup X_P \cup X_D$ express widget candidates, positioning candidates, and dialogs respectively.**
- **Values of variables $x_{Wi} \in X_W$, $x_{Pj} \in X_P$, and $x_D \in X_D$ are selected candidates.**

FCSP

- **Variables**
- Domains
- Constraints



Definition of domains

- **The values of domains are tuples according to each variable type.**
  - **A domain for widget variable x_{Wi}**

$$D_{x_{W_i}} = \{\langle w_i, \underbrace{\mathrm{w}_{w_i}, \mathrm{h}_{w_i}}\rangle \mid w_i \in W_i\}$$

Minimum size of widget w_i

An example:

$$D_{xw1} = \{<\texttt{check\_box}, 210, 18>,$$
$$<\texttt{radio\_buttons}, 210, 36>,$$
$$<\texttt{drop\_down\_list\_box}, 210, 18>\}$$

– A domain for positioning variable x_{Pi}

$$D_{x_{P_j}} = \{\langle p_j, \underbrace{\mathrm{w}_{p_j}, \mathrm{h}_{p_j}}, M_{p_j}\rangle \mid p_j \in P_j\}$$

Minimum size of positioning p_j

$$M_{p_j} = \langle \underbrace{\mathrm{W}_{p_{j,1}}, \mathrm{H}_{p_{j,1}}}, \mathrm{W}_{p_{j,2}}, \mathrm{H}_{p_{j,2}}, \ldots, \mathrm{W}_{p_{j,K_{P_j}}}, \mathrm{H}_{p_{j,K_{P_j}}} \rangle$$

Permissible maximum size for each child

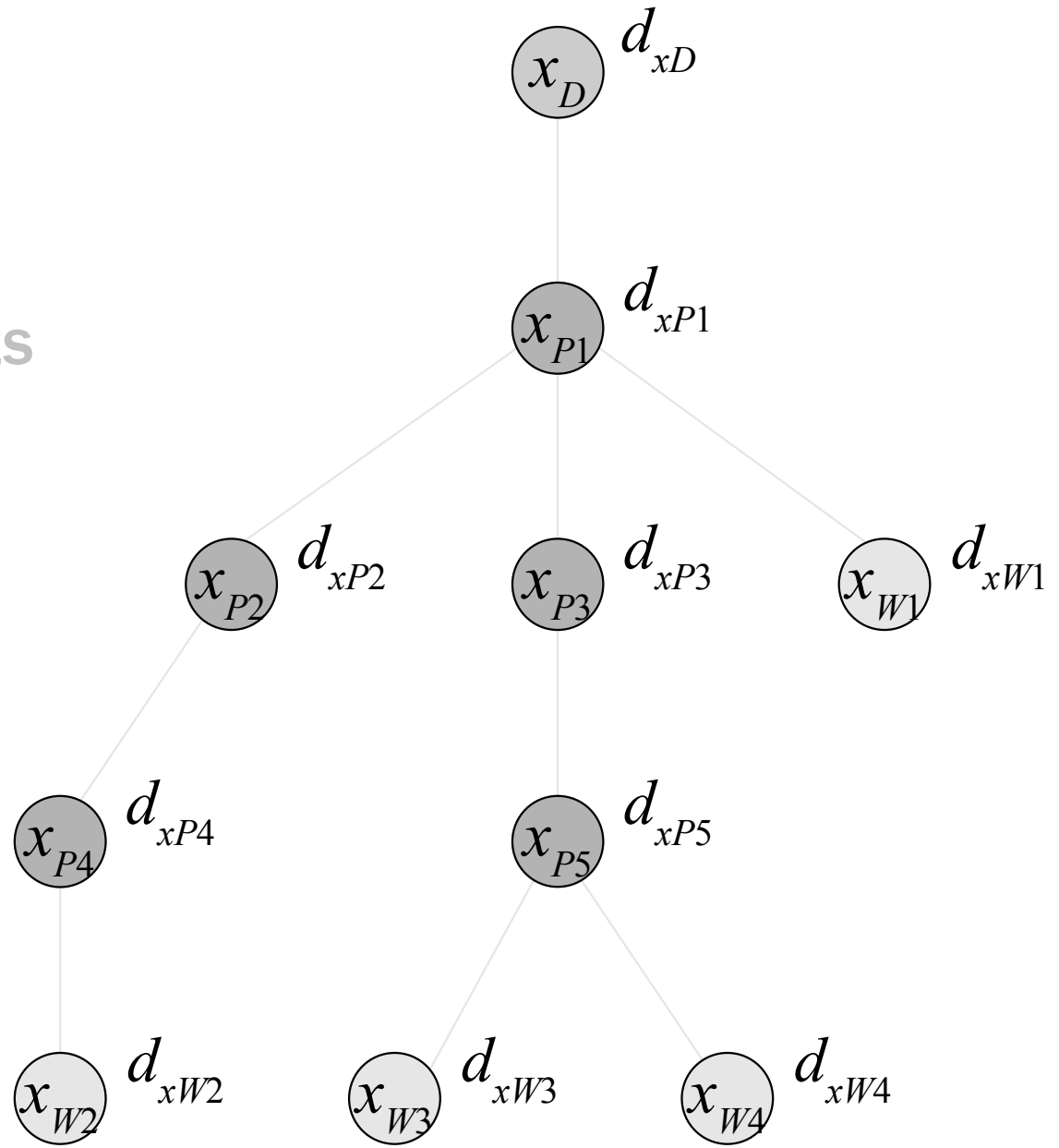
– A domain for dialog variable x_D

$$D_{x_D} = \{\underbrace{\langle \mathrm{W}_d, \mathrm{H}_d \rangle}\}$$

Size of the dialog

FCSP

- Variables
- Domains
- Constraints



Definition of constraints

- Each variable except for a dialog variable is connected to a **unary constraint** for expressing its desirability.

  Satisfaction degree = desirability of candidate

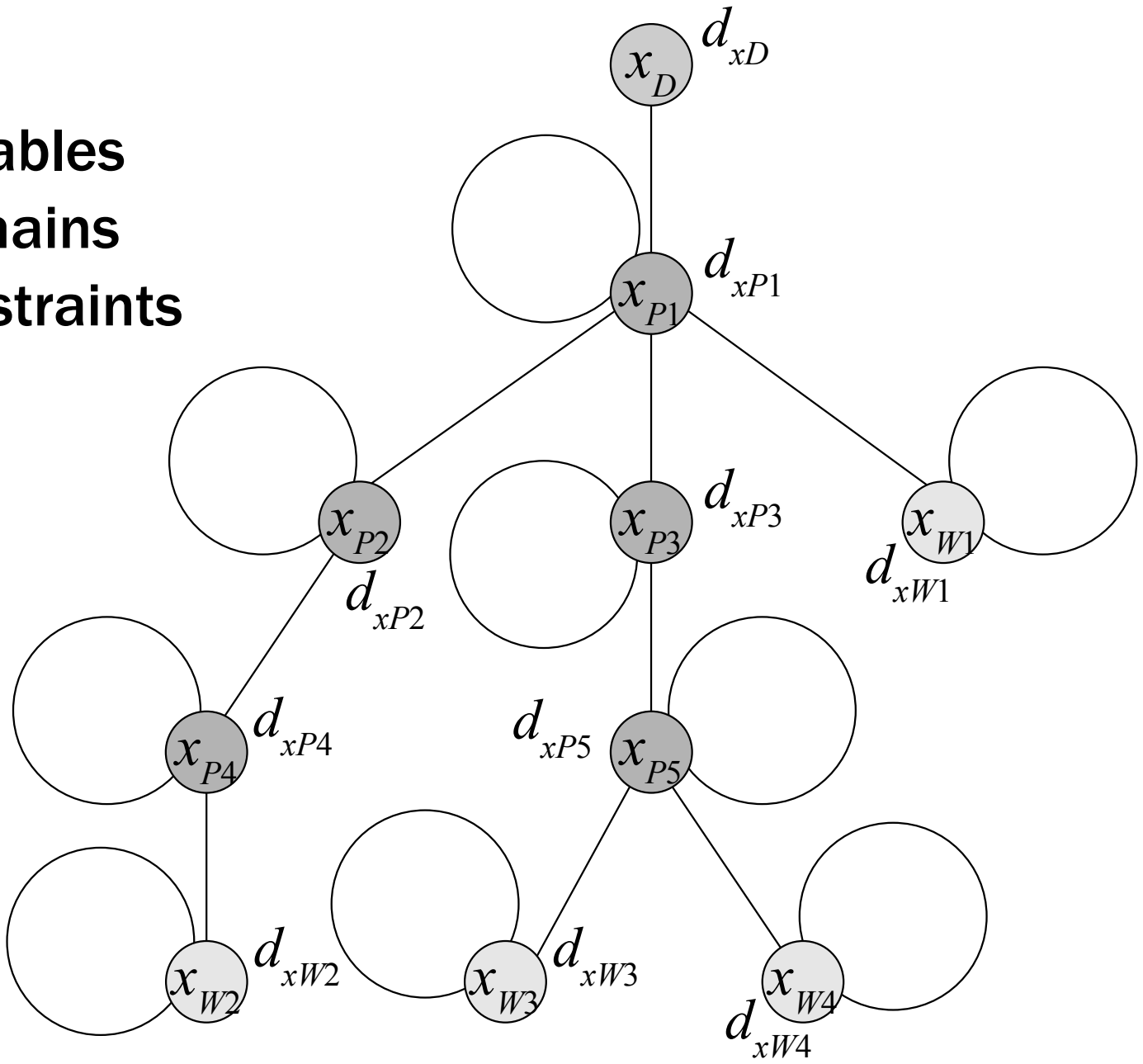
- Two variables of a container and its child are connected to a **binary constraint** for expressing a inclusion relation.

$$c_{con\,P_{j,k}}(v_1, v_2) = \begin{cases} 1 & \text{if } \mathrm{w}_{p_{j,k}} \leq \mathrm{W}_{p_{j,k}} \text{ and } \mathrm{h}_{p_{j,k}} \leq \mathrm{H}_{p_{j,k}} \\ 0 & \text{otherwise} \end{cases}$$

Is the permissible size for a child larger than its minimum size ?

FCSP

- Variables
- Domains
- Constraints



IV. IMPLEMENTATION

Three phases for FWL

- We implemented an experimental system for FWL, which consists of three phases:

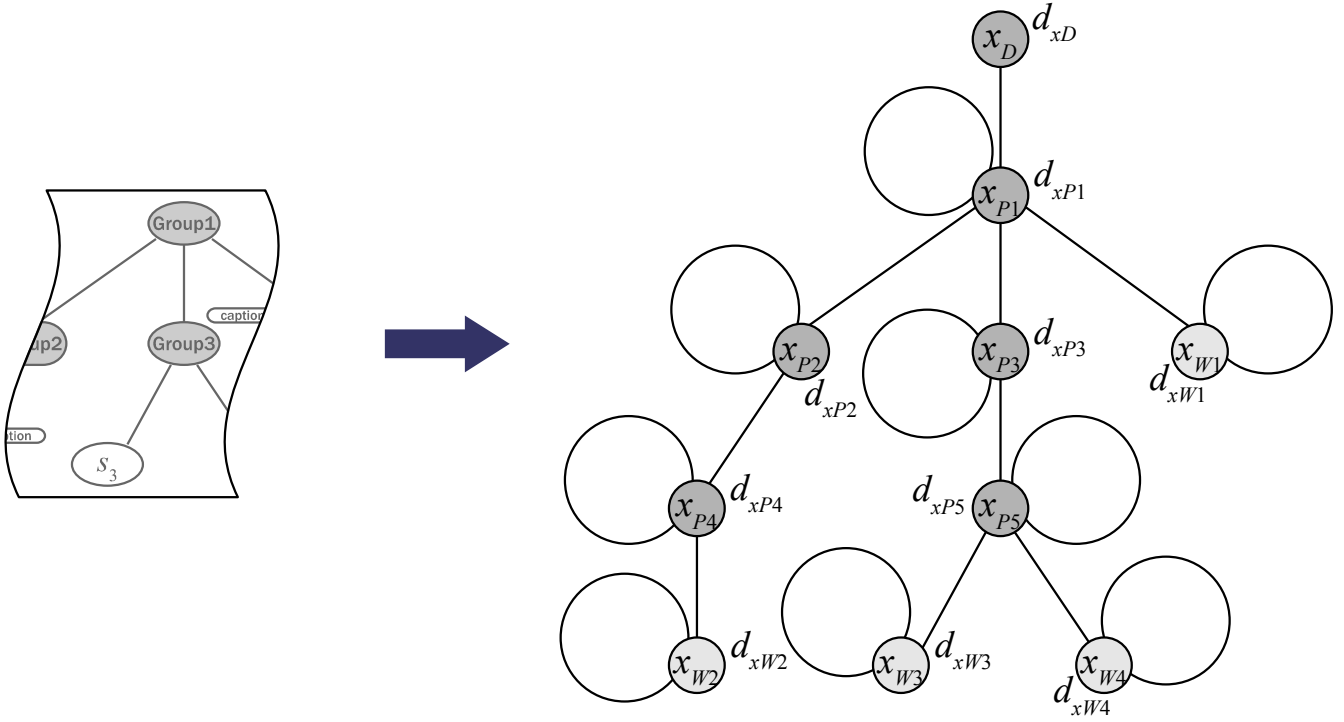
  A) creating a FCSP from a UI model,

  B) solving the problem with an algorithm, and

  C) performing actual layout based on the result of the algorithm.

A. Creating problem phase

- A constraint graph is generated from a given UI model.



Calculate minimum sizes

- The minimum sizes of widgets (the values of the domains of the variables) are decided by the parameters of the selection acts and TABLE 2.

TABLE 2

Widget	Minimum height (without edges)
Check box	$item\_h$
Drop down list box	
List box	$\min(\|L\|, 4)\ item\_h$
Radio buttons	$\|L\|\ item\_h$
Check boxes	

- **The minimum sizes of containers are calculated by the minimum sizes of their child elements.**
  - **Array container**

$$\mathrm{w}_{ac_j} = \begin{cases} \max(\mathrm{w}_{ac_{j,k}}) \\ \sum \mathrm{w}_{ac_{j,k}} \\ \max(\mathrm{w}_{ac_{j,k}}) \end{cases} \qquad \mathrm{h}_{ac_j} = \begin{cases} \sum \mathrm{h}_{ac_{j,k}} & \text{vertical alignment} \\ \max(\mathrm{h}_{ac_{j,k}}) & \text{horizontal alignment} \\ \max(\mathrm{h}_{ac_{j,k}}) & \text{tab pages} \end{cases}$$

  - **Labeled container**

$$\mathrm{w}_{lc_j} = \begin{cases} \mathrm{w}_{lc_{j,1}} + \mathrm{lw}_{lc_j} \\ \max(\mathrm{w}_{lc_{j,1}}, \mathrm{lw}_{lc_j}) \end{cases} \qquad \mathrm{h}_{lc_j} = \begin{cases} \max(\mathrm{h}_{lc_{j,1}}, \mathrm{lh}_{lc_j}) & \text{left side} \\ \mathrm{h}_{lc_{j,1}} + \mathrm{lh}_{lc_j} & \text{upper side} \end{cases}$$

Calculate maximum sizes

- The permissible maximum sizes of child elements are calculated with a dialog size.
  - Array container

$$\mathrm{W}_{ac_{j,k}} = \begin{cases} \mathrm{W}_{ac_j} \\ \mathrm{W}_{ac_j} - \sum_{l \neq k} \mathrm{w}_{ac_{j,l}} \\ \mathrm{W}_{ac_j} \end{cases} \qquad \mathrm{H}_{ac_{j,k}} = \begin{cases} \mathrm{H}_{ac_j} - \sum_{l \neq k} \mathrm{h}_{ac_{j,l}} & \text{vertical} \\ \mathrm{H}_{ac_j} & \text{horizontal} \\ \mathrm{H}_{ac_j} & \text{tab} \end{cases}$$

  - Labeled container

$$\mathrm{W}_{lc_{j,1}} = \begin{cases} \mathrm{W}_{lc_j} - \mathrm{lw}_{lc_j} \\ \mathrm{W}_{lc_j} \end{cases} \qquad \mathrm{H}_{lc_{j,1}} = \begin{cases} \mathrm{H}_{lc_j} & \text{left} \\ \mathrm{H}_{lc_j} - \mathrm{lh}_{lc_j} & \text{upper} \end{cases}$$

B. Solving problem phase

- The system iterates steps of solving the generated FCSP with the forward checking algorithm looking for a better solution.
- The system prunes the domains according to a worst satisfaction degree.

- Step 1
  - The system makes a satisfaction degree set by collecting possible degrees from all unary constraints.
- Step 2
  - The system chooses a maximum from the set, and sets it as the worst satisfaction degree.
  - It prunes values of the domains whose satisfaction degree of the unary constraints are less than the worst satisfaction degree.

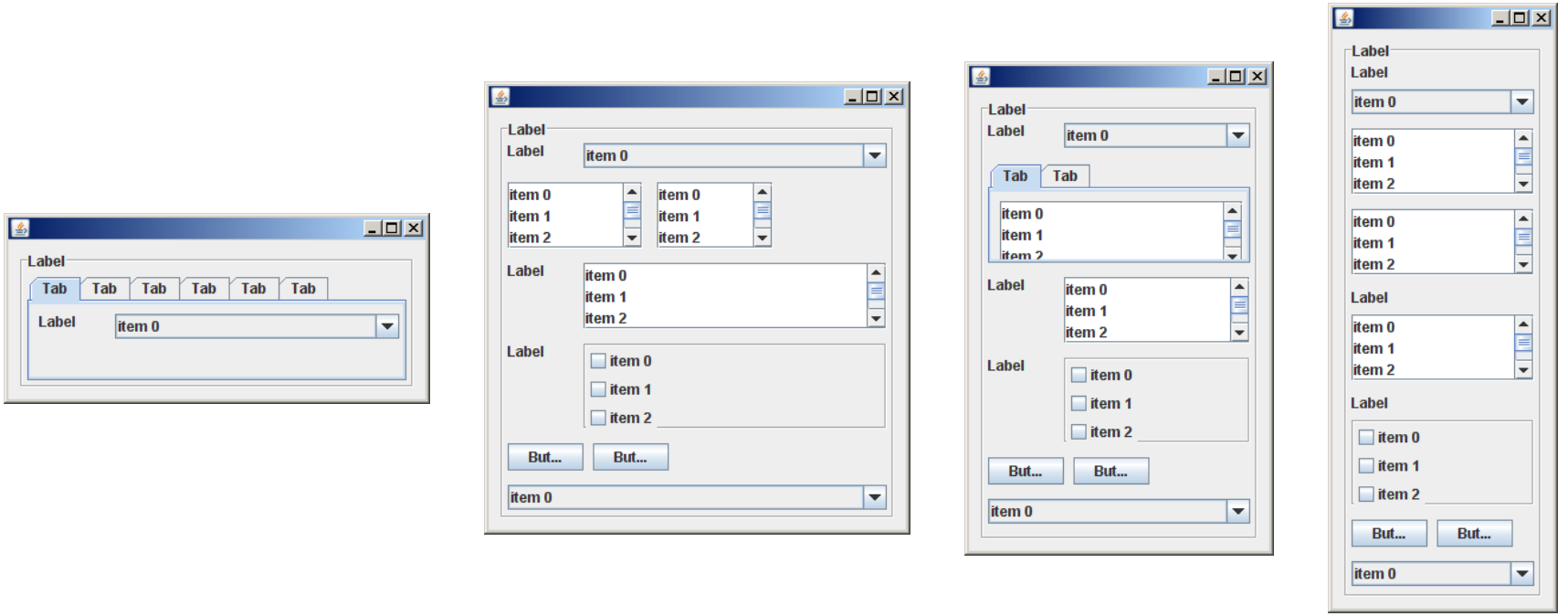
- Step 3
  - The system solves the FCSP with the forward checking algorithm, which is extended for handling fuzzy problems.
- Step 4
  - If the system can find a solution, it moves to the next phase in order to do an actual layout;
  - Otherwise, it moves back to the step 2, or it stops in failure if no value remains in the set.

Pruning

- The pruning of domains are done before applying the algorithm for solving the problem rapidly.
- The forward checking algorithm guarantees that it finds a solution if one exists, but it has a disadvantage that it requires large time.
- Hence, it is effective to reduce the scale of the problem by the pruning.

C. Layout with result phase

- Based on an assignments of variables, the system decides positions and sizes of the selected widgets, and it places them.

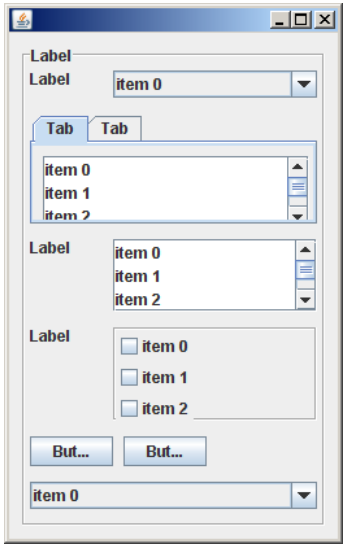


* Dialog boxes generated automatically from the same description

V. DISCUSSION

Speed of doing layout

- We have confirmed that it can finish performing the layout of the example fast enough for GUI generation.
  - 250 msec (without pruning, more than 50000 msec)
  - Environment:
    - Pentium M 1.10 GHz CPU
    - 512 MB memory
    - Windows XP Professional edition
    - Java 6 SE



How to define variables

- In the early stage, we tried to formulate FWL problems with variables expressing widgets sizes and positions, but we were not able to obtain enough speed for solving it.
- That is because the variables have large domains, and the scale of the problem is enlarged.

VI. CONCLUSION AND FUTURE WORK

Conclusion

- We have formulated the layout problem accompanied by widget selections, named the flexible widget layout problem, as a fuzzy constraint satisfaction problem.
- We have offered the solution solving it in a practical time for users.

Future work

- We need to
  - add some layout rules based on GUI guidelines,
  - evaluate the relation between problem scales and solving times, and
  - consider other algorithms for FCSPs.