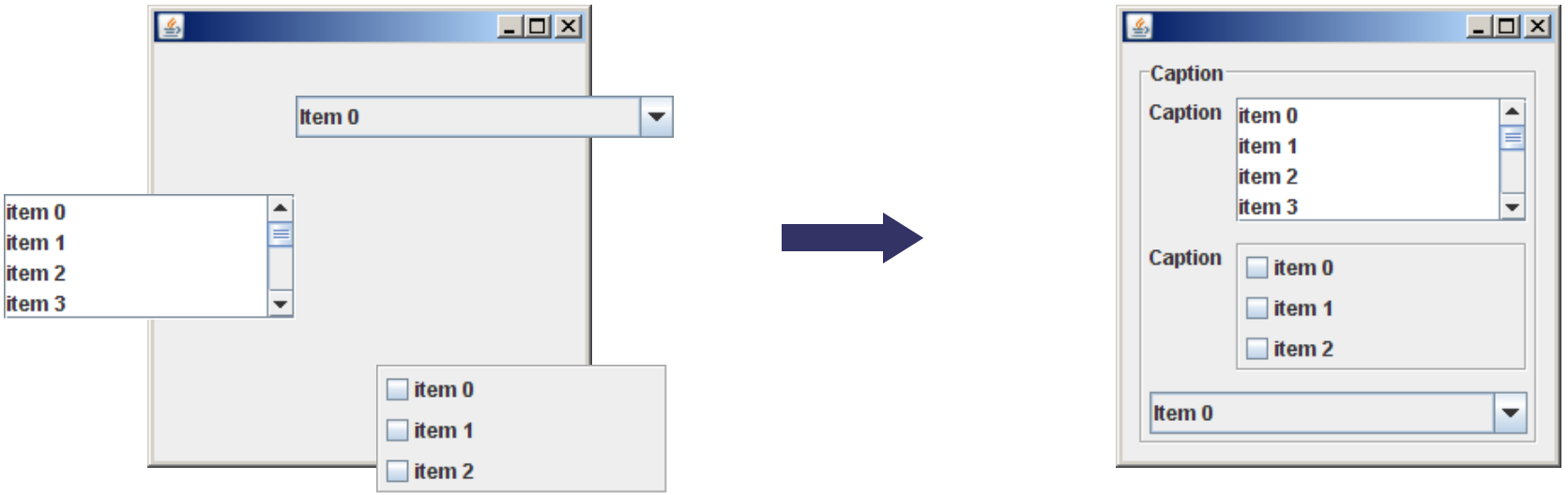
Flexible Widget Layout Formulated as Fuzzy Constraint Satisfaction Problem

Takuto Yanagida and Hidetoshi Nonaka

Hokkaido University, Japan

Widget layouts

- The process of deciding positions and sizes of widgets (list boxes, radio buttons, and panels)



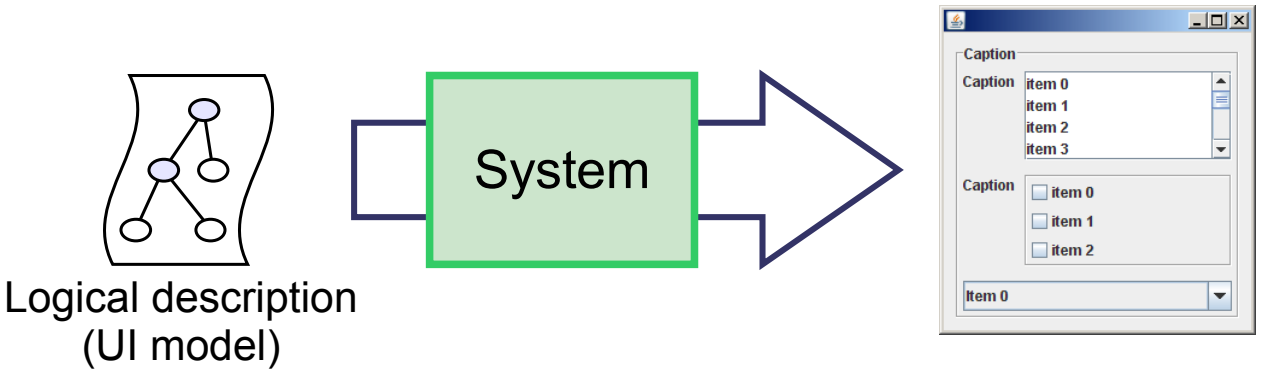
The layout has a significant impact on the usability of tasks which can be accomplished with GUIs.

Model-based UI design

- In the field of model-based UI design
  - Systems generate GUIs from logical descriptions.

> **Logical descriptions** (UI models)
>
> - specifying **UI functions** independently of platforms, instead of specifying widgets.

  - It is useful for realizing the diversity of UIs.



Logical description
(UI model)

Widget layouts + model-based UI

- In the field of model-based UI design
  - A layout system needs **to select widgets** before making a layout.
  - In addition, widgets are sometimes not uniquely determined.

A system could select small widgets with little usability for small screens, or large ones with enough usability for large screens.

→ Related studies on how to generate GUIs

*Related work (1/2)

- Design time layout system
  - An adaptive algorithm for automated UI design [a]
  - An approach using mathematical relationships [b]
- Dynamic layout
  - GADGET [c]
  - SUPPLE [d]

[a] J. Eisenstein, A. Puerta, and R. Software. Adaption in automated user-interface design. In Proc. of IUI 2000, 2000.
[b] F. Bodart, A.-M. Hennebert, J.-M. Leheureux, and J. Vanderdonckt. Towards a dynamic strategy for computer-aided visual placement. In Proc. of AVI ’94, pp. 78–87, Italy, 1994.
[c] J. Fogarty and S. E. Hudson. Gadget: a toolkit for optimization-based approaches to interface and display generation. In Proc. of UIST ’03, pp. 125–134, Canada, 2003.
[d] K. Gajos and D. S. Weld. SUPPLE: automatically generating user interfaces. In Proc. of IUI ’04, pp. 93–100, Portugal, 2004.

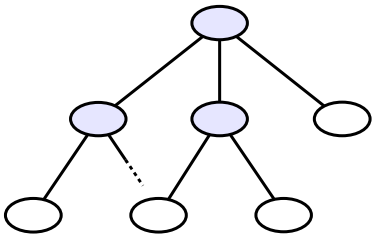
*Related work (2/2)

- Plasticity of widgets
  - Handling widget selections as plasticity [e]
  - The graceful degradation [f]
  - An intelligent editor for GUIs [g]
- Other studies
  - Many studies for the LSI or VLSI layout problem
  - Existing *layout managers* offered by GUI toolkits

[e] G. Calvary, J. Coutaz, D. Thevenin, Q. Limbourg, L. Bouillon, and J. Vanderdonckt. A unifying reference framework for multi-target user interfaces. Interacting with Computers, 15:289–308, 2003.
[f] M. Florins and J. Vanderdonckt. Graceful degradation of user interfaces as a design method for multiplatform systems. In Proc. of IUI 2004, pp. 140–147, Portugal, 2004.
[g] B. Collignon, J. Vanderdonckt, and G. Calvary. An intelligent editor for multi-presentation user interfaces. In Proc. of SAC 2008, pp. 1634–1641, Brazil, 2008.

Consideration (1/3)

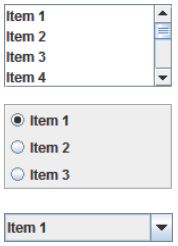
- GUI generations in model-based UI designs
  - How to generate GUIs from UI models?



How?

How to **select widgets** corresponding to UI functions?

F



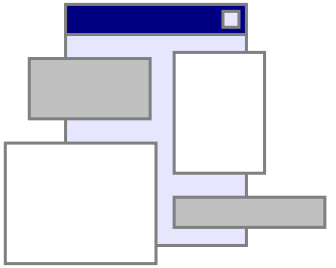
Which?

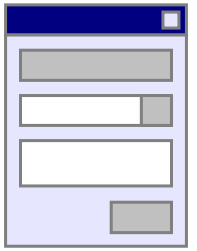
A UI function

Widget candidates

Consideration (2/3)

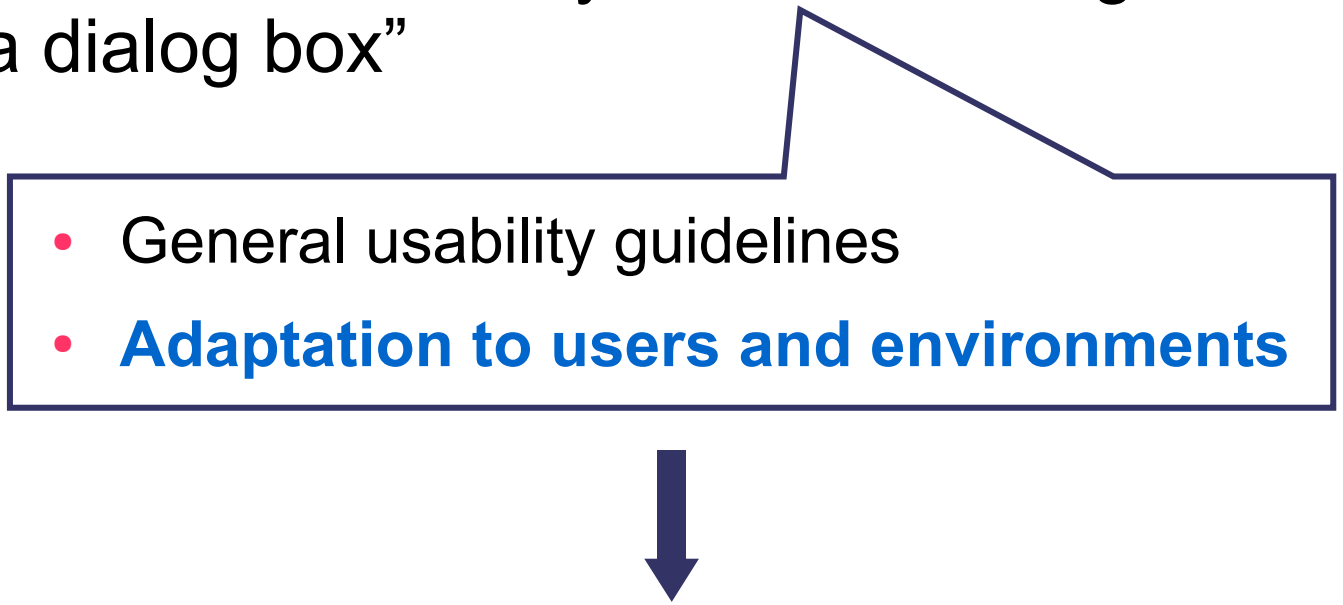
- Viewpoint of **desirability**
  - General usability guidelines
  - **Adaptation to users and environments**
- Tactics of widget selections
  1. ~~To select based on **ONLY** desirability~~
     - A tendency that larger widgets are more usable
  2. To select moderately desirable ones
     - All widget can be put in the inside of a dialog box





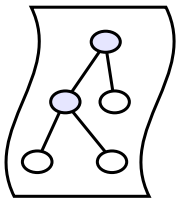
Consideration (3/3)

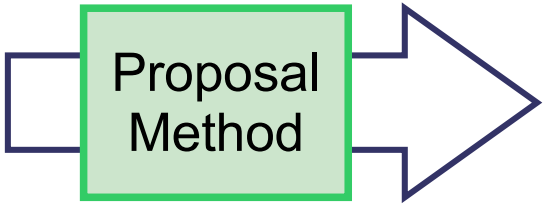
- GUI generations in model-based UI designs
  - “To select moderately **desirable** widgets to be put in a dialog box”

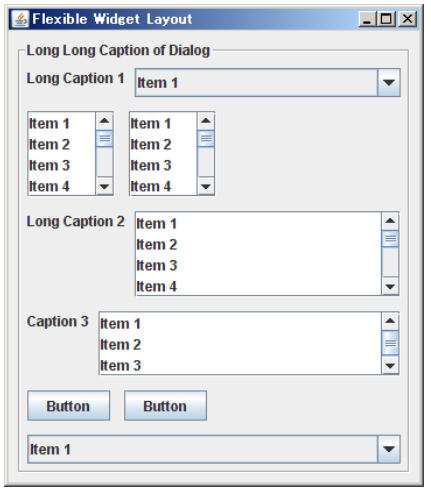


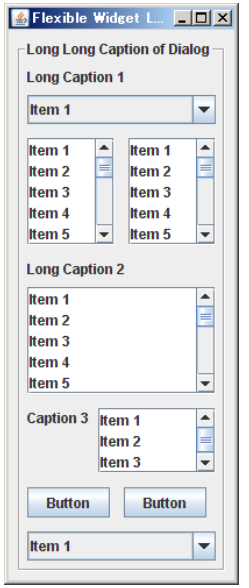
Objective (1/2)

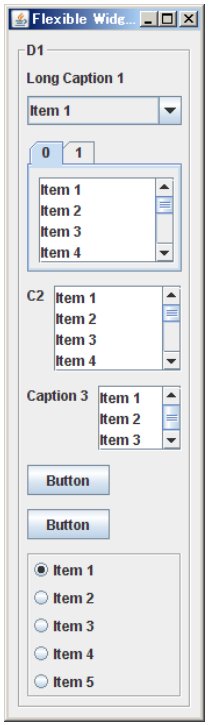
- Flexible Widget Layout (FWL)
  - Automated GUI generation based on UI models
    - Where widgets to be used are dynamically selected,
    - Layout processes are rapidly finished.











GUIs corresponding to the same UI model

Objective (2/2)

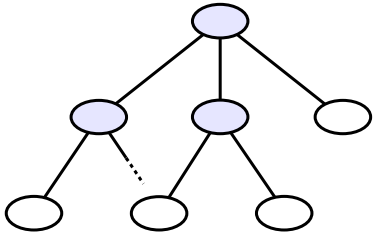
- Flexible Widget Layout Problem
  - Widget selections
  - Desirability of layout

  - Combinatorial searches
  - Fuzzy constraints

- Formulation as **fuzzy constraint satisfaction problems** (FCSPs)
  - Combinatorial search problem that decides assignments to variables that satisfy all constraints among variables

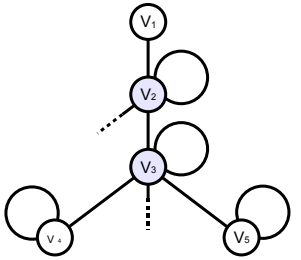
Phases of FWL

1. Generate an FCSP from given UI model



UI model

Phase 1



FCSP

Phase 2

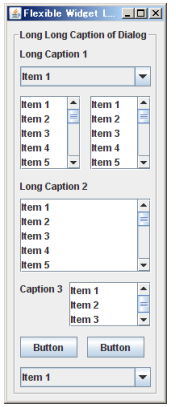
2. Solve the FCSP to get combinations of widgets

$V_1 = 3$
$V_2 = 1$
$V_3 = 0$
$V_4 = 3$
$V_5 = 2$
⋮

Assignments

Phase 3

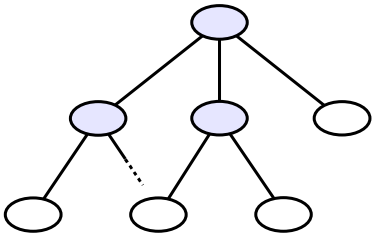
3. Make a layout



Layout

Logical description (UI model)

- Abstract interaction description language (AIDL)
  - UI function description based on selection act model
    - Selection elements (acts)
      - Choices
        - A set of choices
        - A type
    - Group elements
    - Description elements (text)



FWL (1/2)

- Flexible Widget Layout Problem
  1. To determine widget candidate sets
     - Mapping to each element of selection act model
  2. To select widget from each candidate sets
     - Combinatorial search problem of widgets

  - Properties of widgets
    - Minimum size: $ms_w = \langle ms.width_w, ms.height_w \rangle$
    - **Desirability** for each type: $0 \leq \alpha \leq 1$

      ↳ You can define it for each user (adaptation)

FWL (2/2)

- Constraints of layouts (layout rules)
  - Feasibility of Layout
    - Whether or not all widgets can be in a dialog box?

      → Must be satisfied
  - Desirability of layout
    - Minimum of desirability of each selected widgets

      → To be maximized as much as possible

"A layout-able and desirable **solution**"

∎

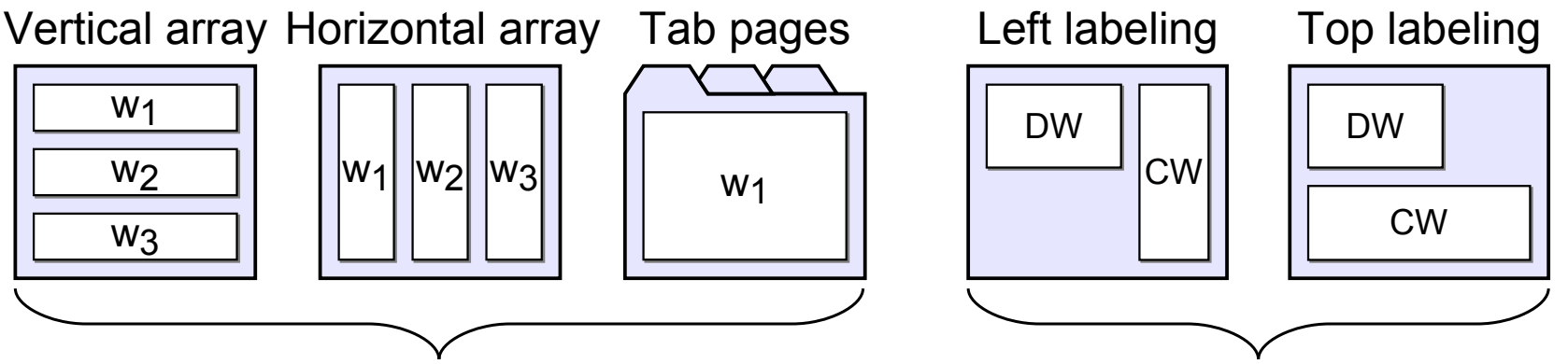
A combination of widgets

Widgets (1/3)

- Container widgets
  - **Selection of container widgets** express selection of positioning.
  - Group elements and *positioning of description*

⟷ Container widget candidate set $W_i \subset W_C$

$W_C =$



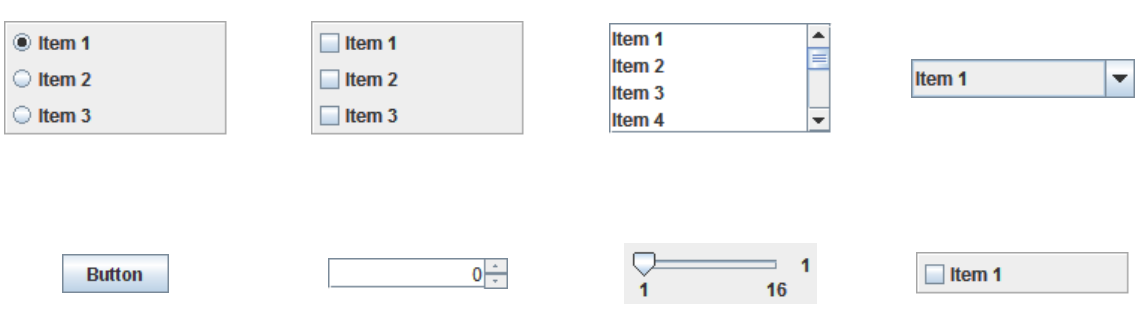
Widgets (2/3)

- Normal widgets
  - A subset adopted in many toolkits (8+2 types)
  - For selection and description elements

⟷ Normal widget candidate set $W_i \subset W_N$

$W_N =$

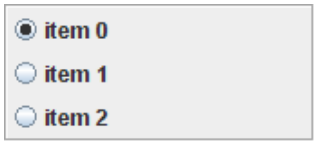
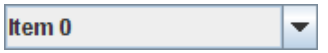
For selection elements



For description elements { Caption labels, Abbreviation labels

Widgets (3/3)

- Trade-off between usability and the ease of layout

	Radio buttons		Drop-down list box
Appearance			
UI function	Same		Same
The ease of layout	Worse		Better
Usability	Better		Worse

FCSP (1/2)

- **Fuzzy** constraint satisfaction problem (FCSP)
  - A simple model for formulating problems
    - a set of variables $X = \{x_1, \ldots, x_m\}$
    - a set of domains $D = \{D_1, \ldots, D_m\}$
    - a set of constraints $C = \{c_1, \ldots, c_r\}$
  - c_k denotes **membership function** $\mu R_k(v[S_k])$
    - S_k: scope (variables related to c_h)
    - v: assignment for all variables
    - A membership value: **satisfaction degree**

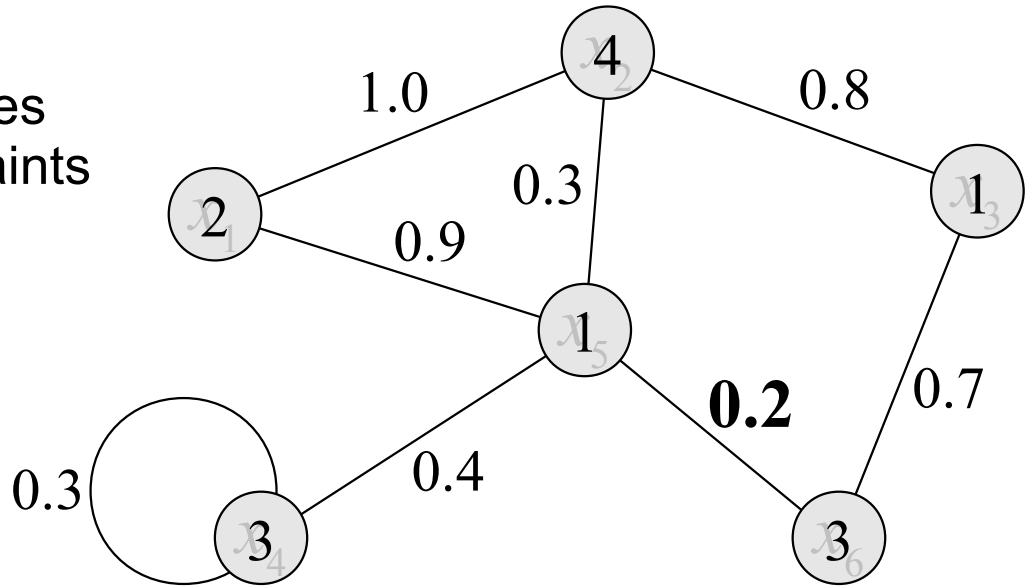
FCSP (2/2)

- A solution of an FCSP
  - 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.

Nodes: variables
Edges: constraints



A solution with satisfaction degree 0.2.

Formulation (1/3)

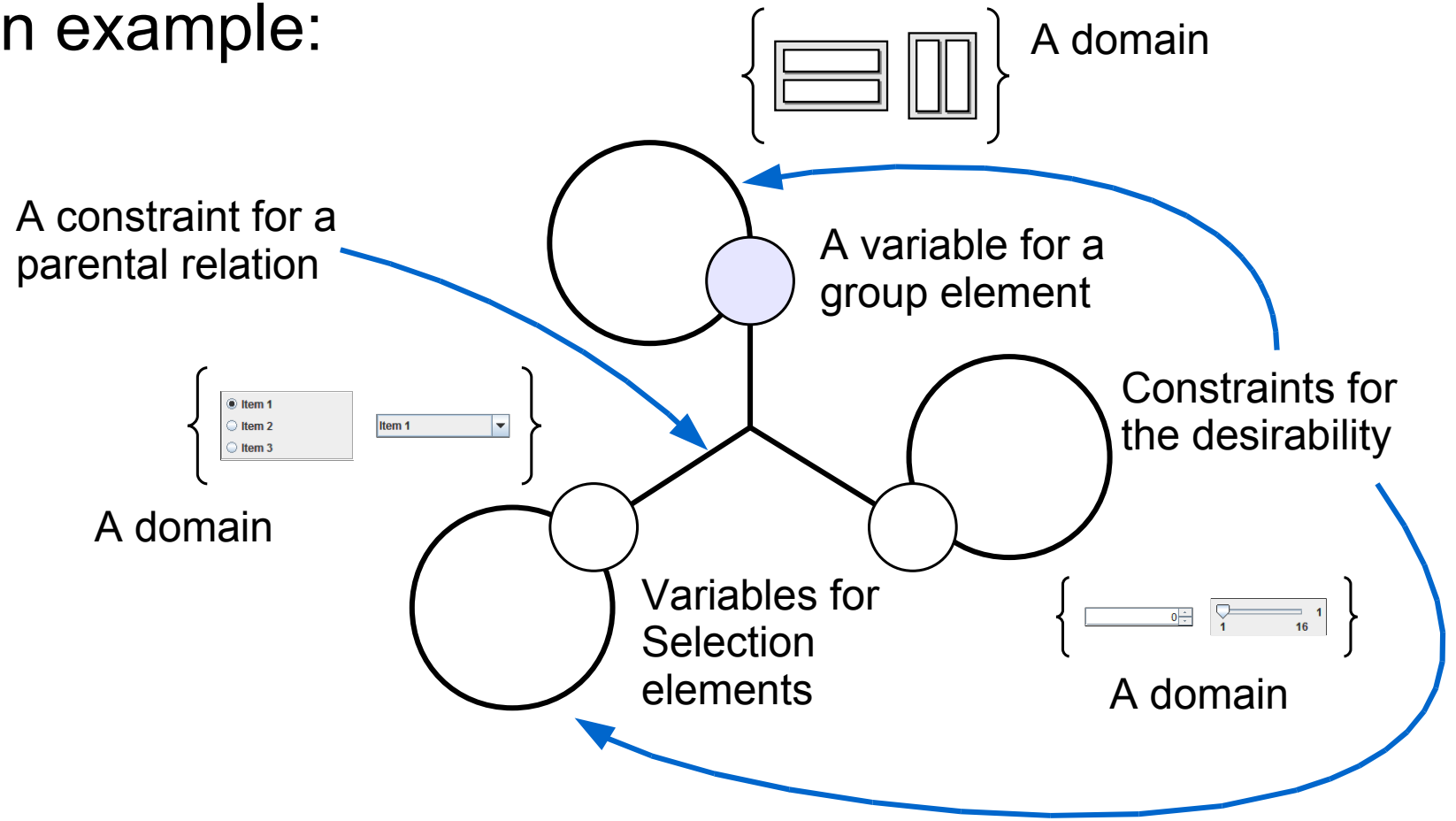
Sizes and positions of widgets are **NOT** represented as variables.

- Variables: selection by its assignment
- Domains: sets of widget candidates
- Constraints : desirability and parental relations

The scale of domains is reduced.

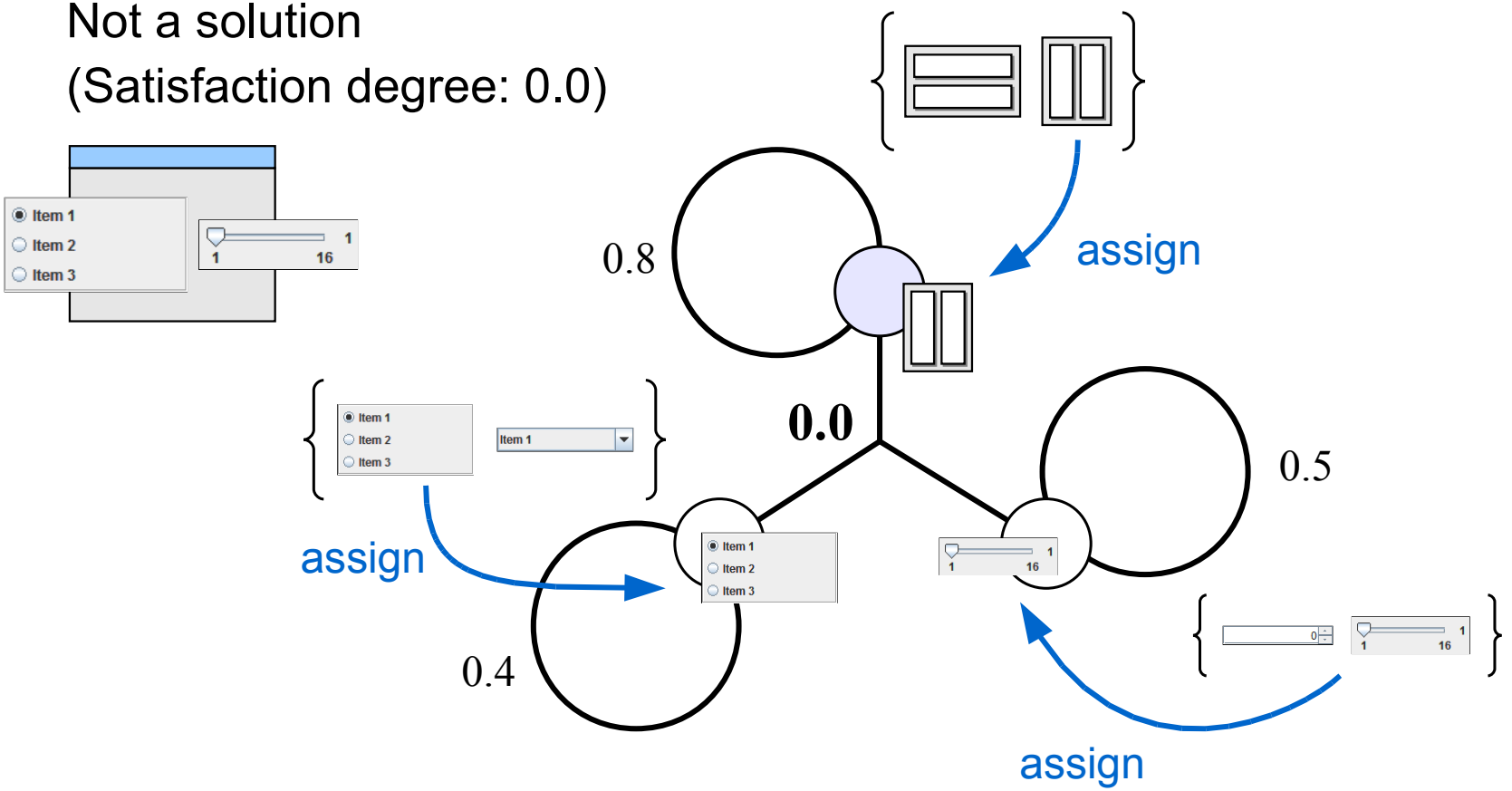
Formulation (2/3)

An example:



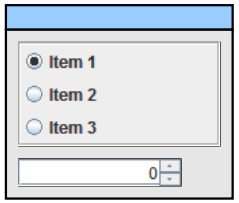
Formulation (2/3)

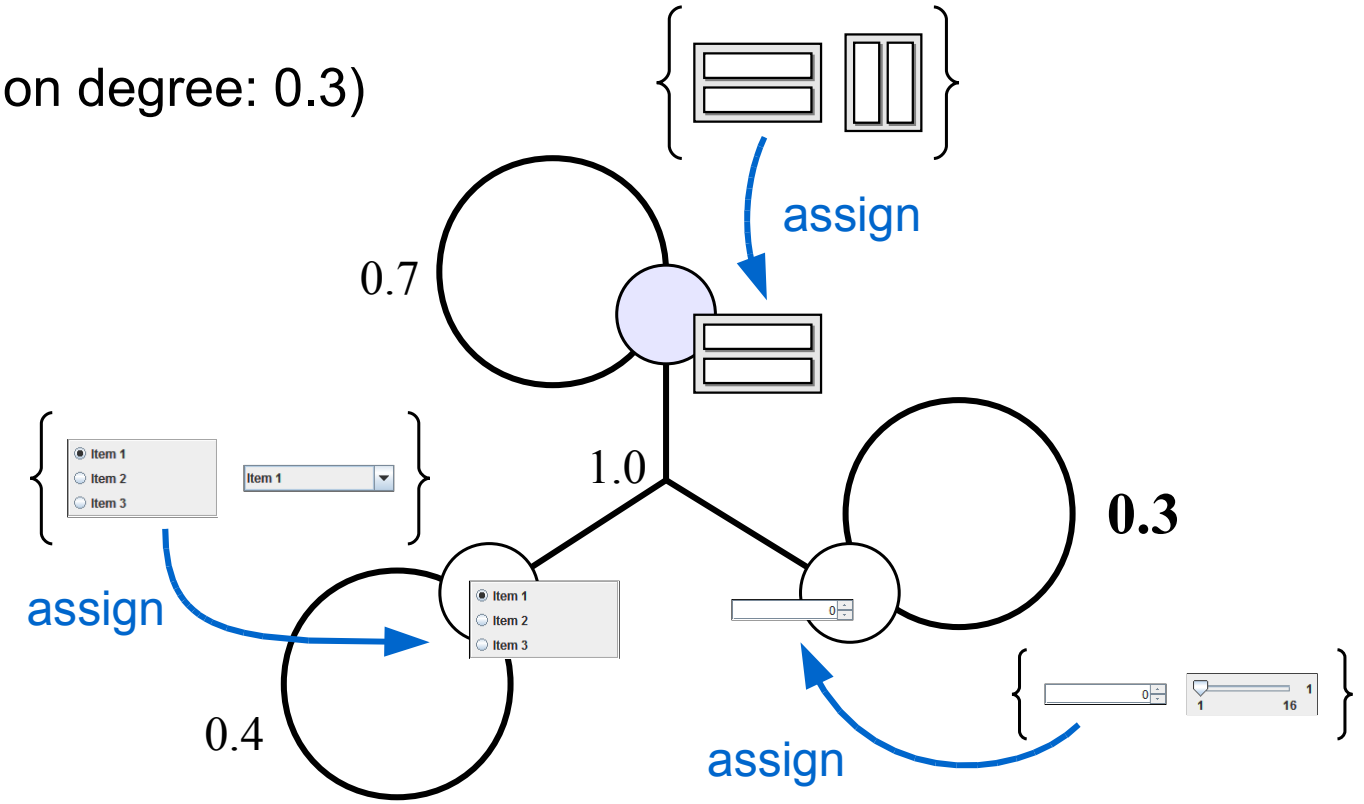
Not a solution
(Satisfaction degree: 0.0)



Formulation (2/3)

A solution
(Satisfaction degree: 0.3)





In practice, constraints for parental relations are binarized for the ease of applying solvers.

Formulation (3/3)

- Changes from the previous work:
  - The formulation is improved so that it coincides more strictly with FCSP.

It enables us to apply various FCSP solvers to the FWL. (in future work)

Demonstration

The FWL system

(An implementation in Java)

* Demo of Flexible Widget Layout
http://aiwww.main.ist.hokudai.ac.jp/~takty/demo/fwl.en.html

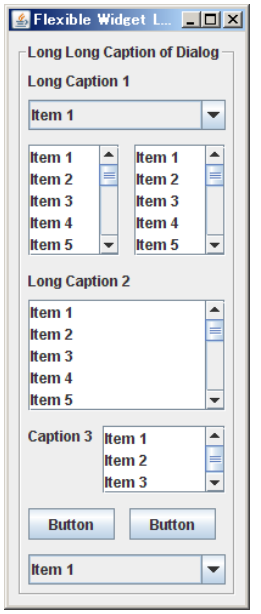
What the system did now

- I changed the size of the dialog box.
- The FWL system
  1. makes an FCSP corresponds to a UI model with pruning domains based on the size,
  2. applies an FCSP solver to the FCSP to obtain a solution,
  3. decides sizes and positions of selected widgets.

*Speed of layout (1/2)

- Preliminary experiment
  - Relation between complexity and time
    'Can we get enough speed?'
    - Environment
      - Java 6
      - Windows XP
      - Turion 64 (2.0 GHz)
      - Desirability defined empirically
    - Condition
      - Change of complexity (+1 to +3)

        (added selection elements to the sample model)

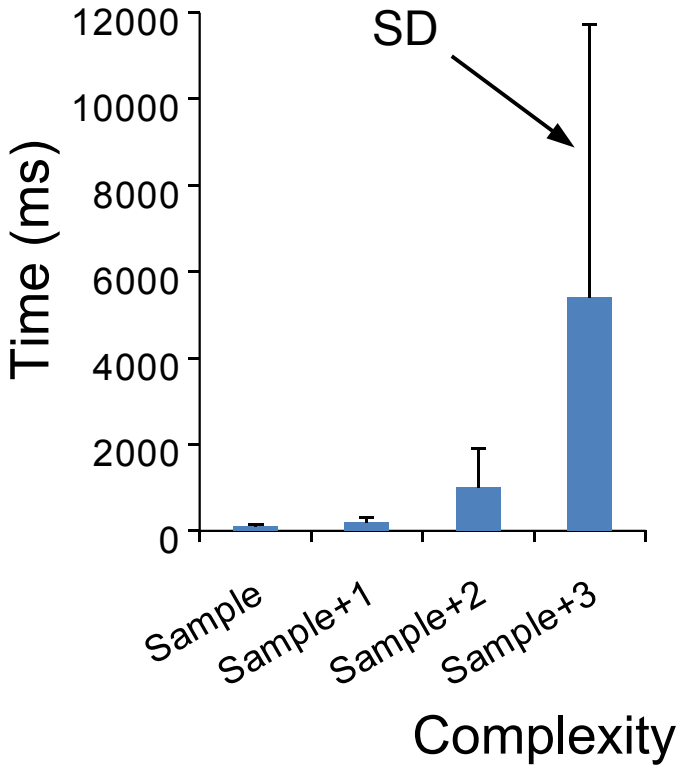


Sample

*Speed of layout (2/2)

- Results
  - Dependence of layout time
    - on **complexity** of models
    - on window sizes
  - Average time: 103ms (sample)
    - Fast enough as the generation time of UIs

      (generally, 1000ms is the rough standard users do not feel waiting)



Conclusion

- The improved formulation of the FWL
  - The formulation coincides more strictly with the **FCSP** framework.
  - We increased the possibility of extending this work with other techniques for **FCSP**.
- Future work
  - to add other layout rules,
  - to evaluate the relation between scales and times,
  - to apply various FCSP solvers.