

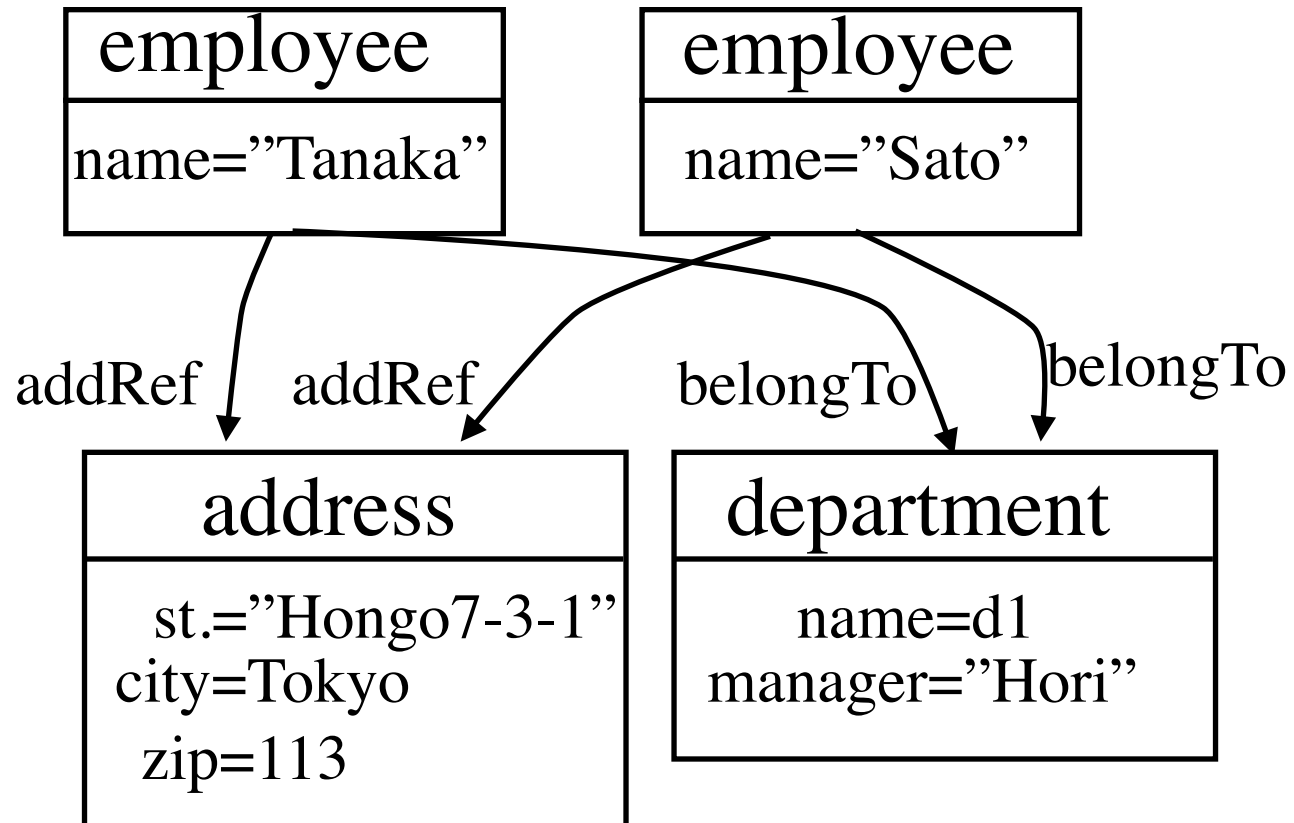
Two Semantics of Updating Graphs

Hiroyuki Kato
National Institute of Informatics, Japan

4th Bi-Trans in ABC

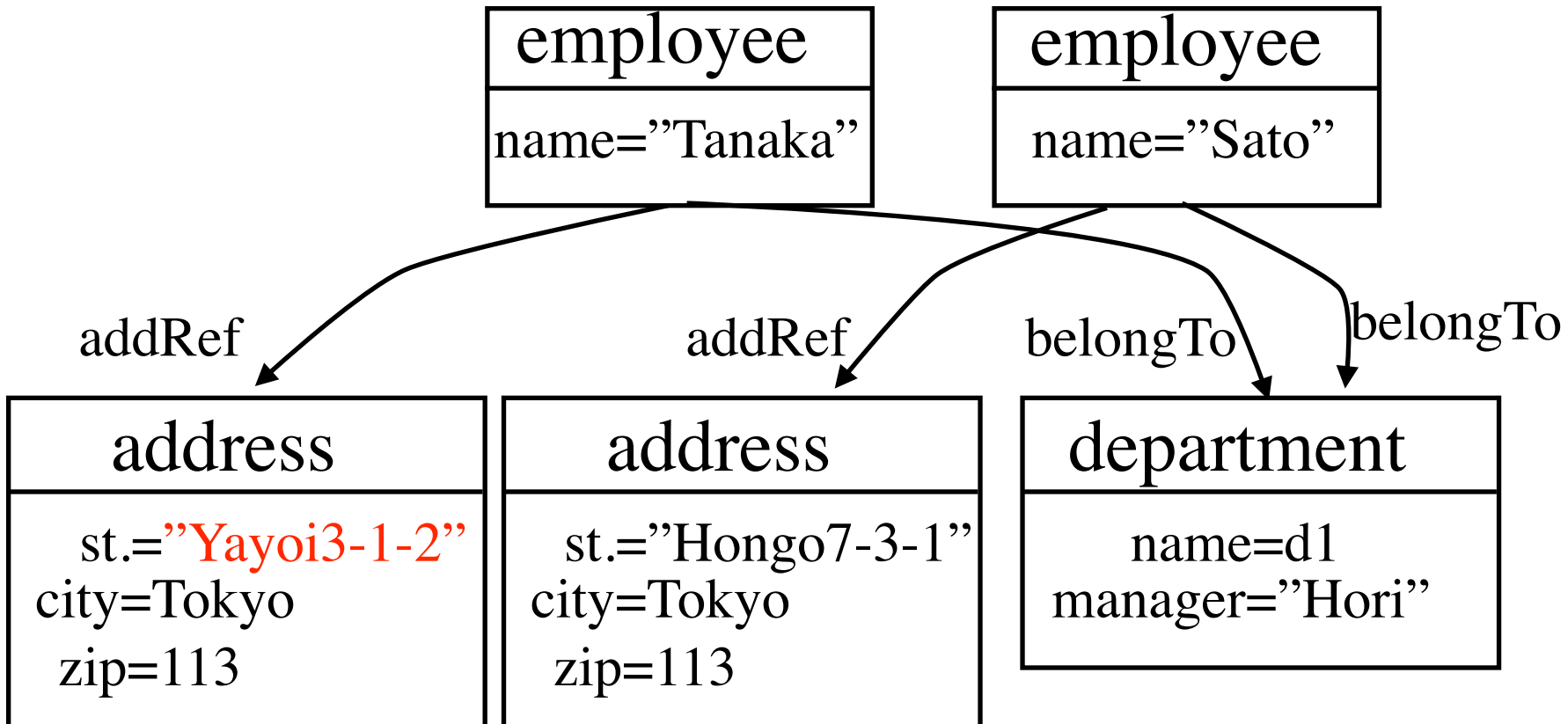
Graphs

- Natural representation of entities in real world



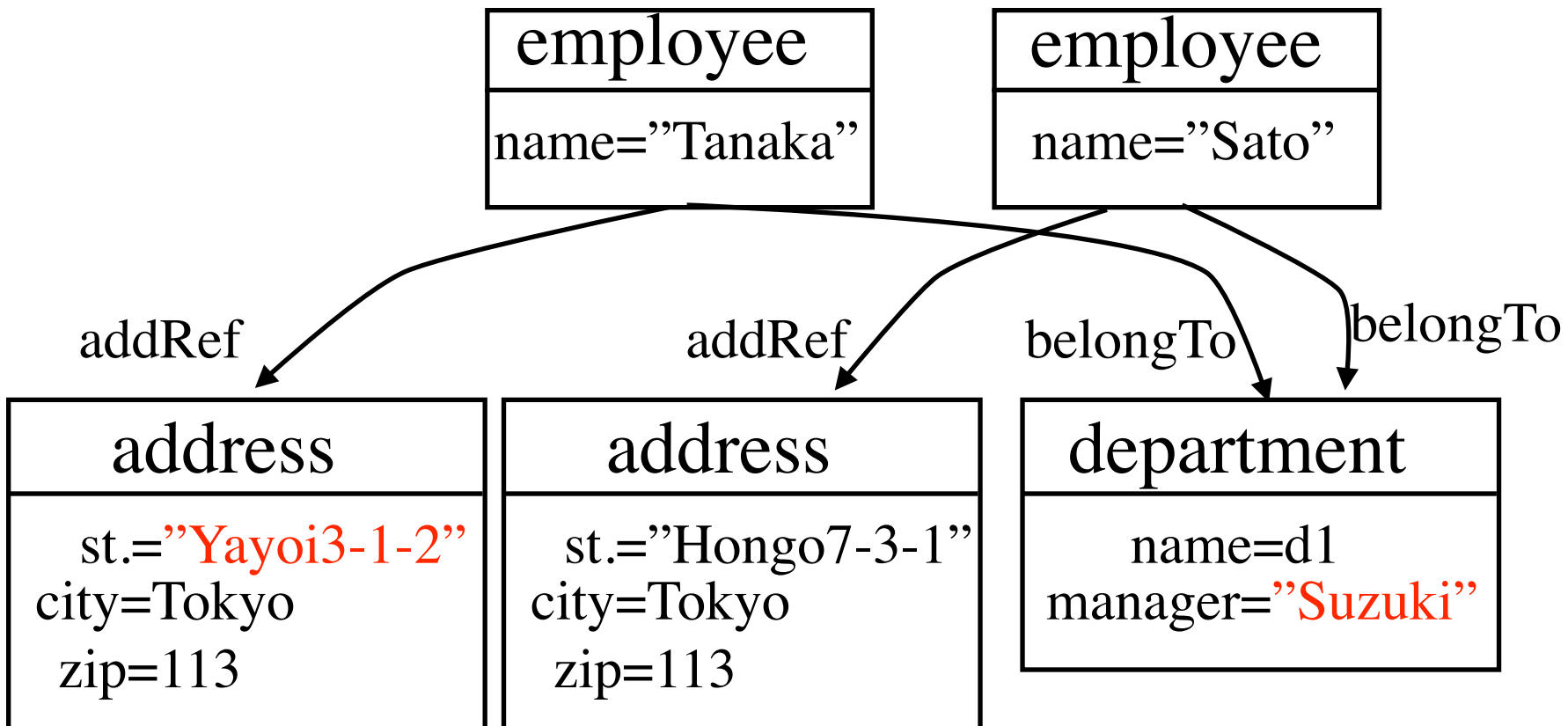
Graphs

- “Tanaka” has moved to new address.



Graphs

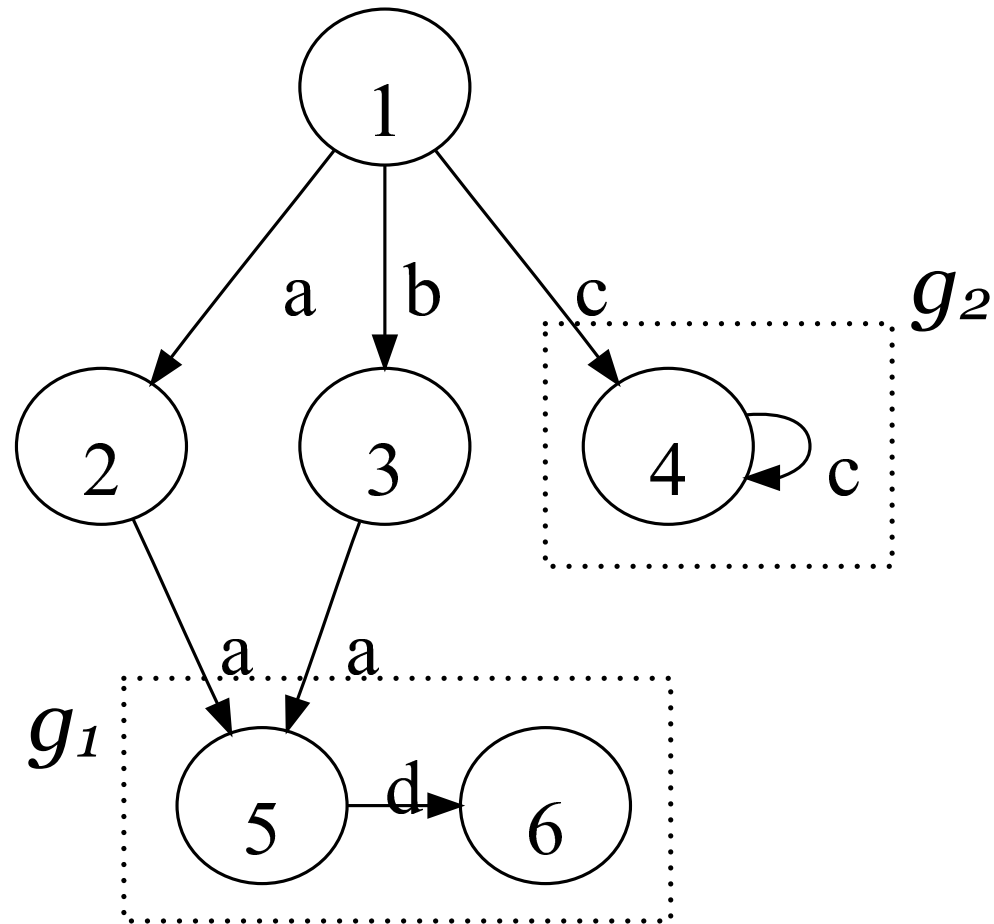
- The manager of d1 has changed.



Outline of this presentation

Towards handling these **two** kinds of update on **shared-nodes** in **our graph model**.

Rooted, Edge-labelled, Set-based, Graphs

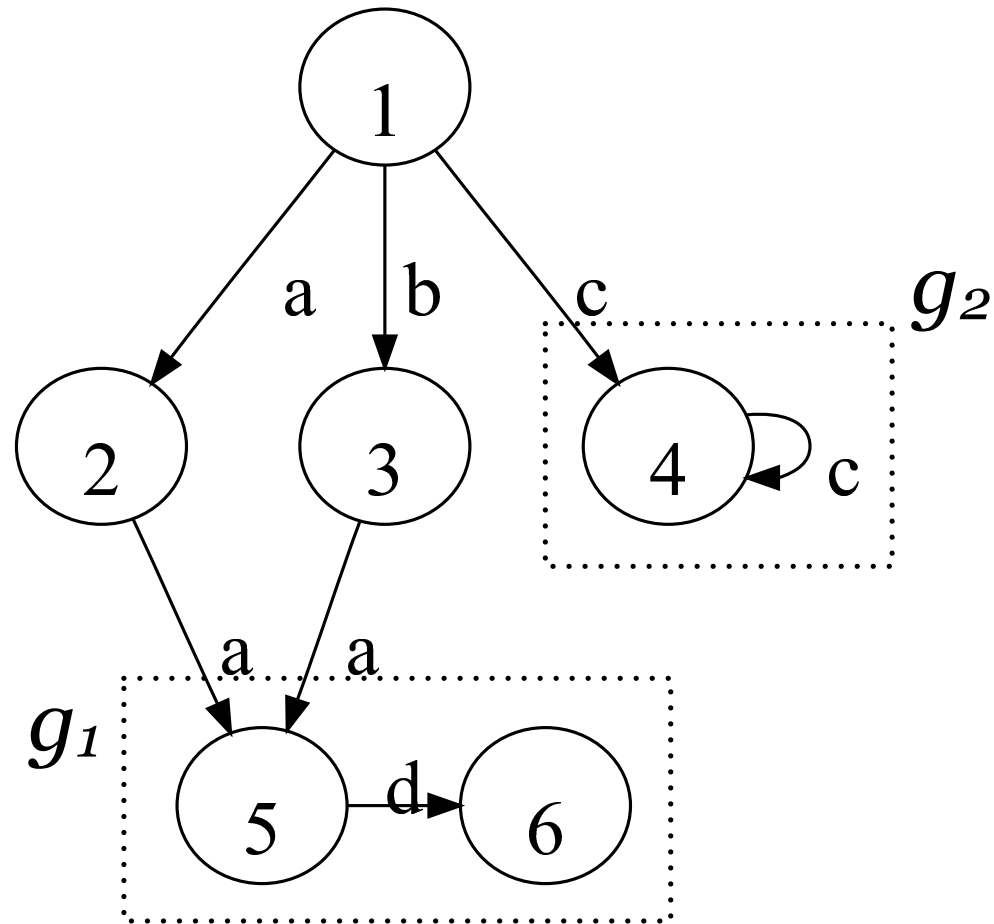


$$g = \{a : \{a : g_1\}, b : \{a : g_1\}, c : g_2\}$$

$$g_1 = \{d : \{\}\}$$

$$g_2 = \{c : g_2\}$$

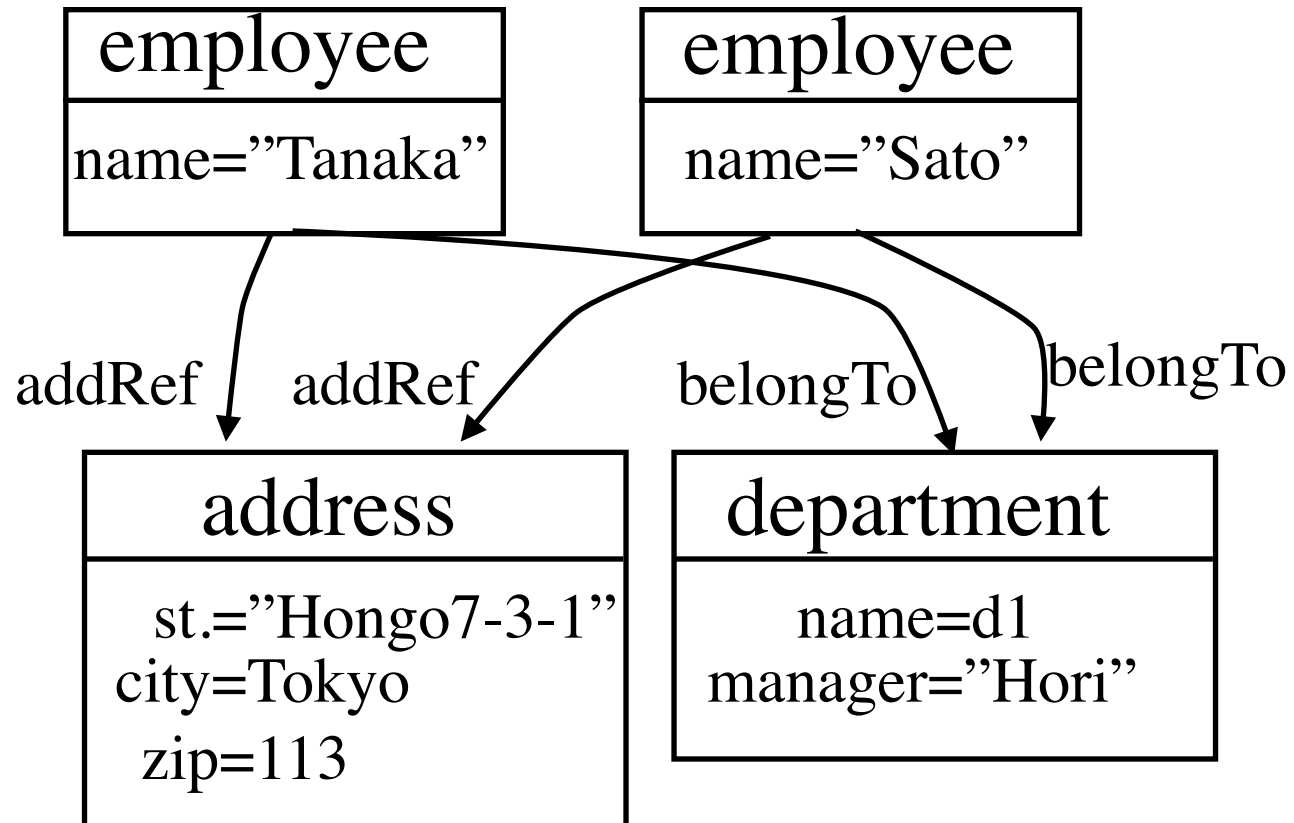
Rooted, Edge-labelled, Set-based, Graphs



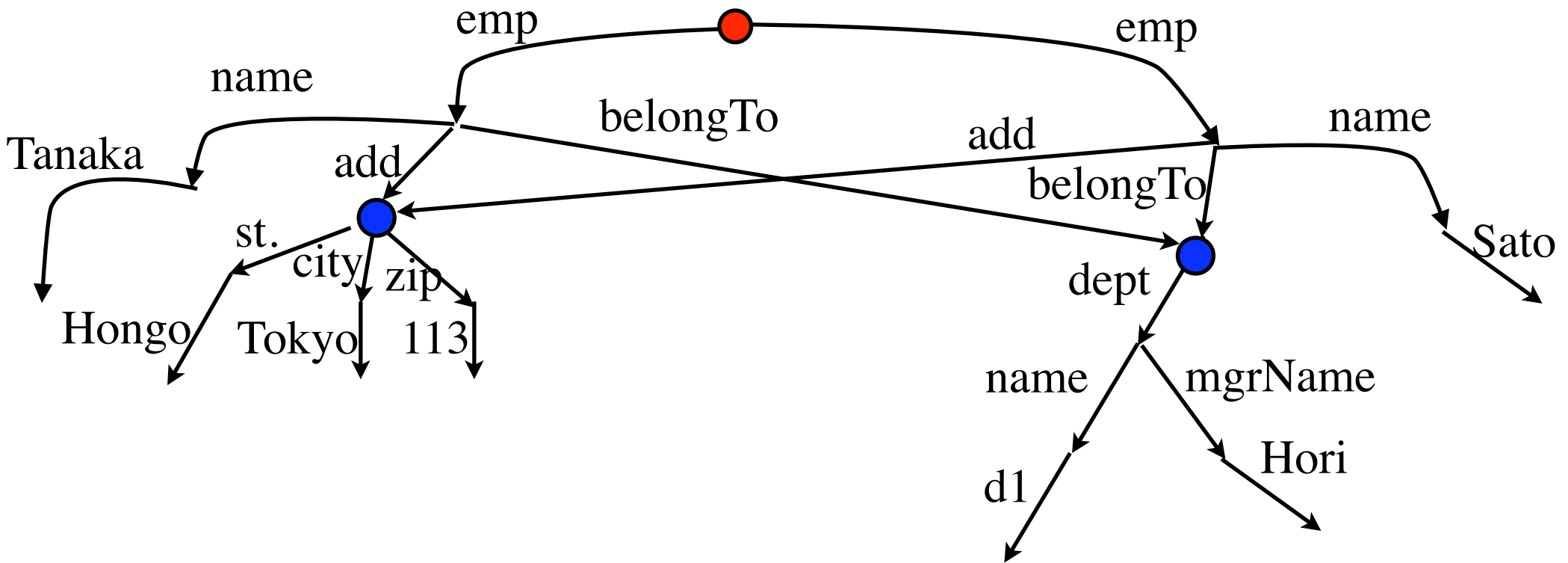
$$g = \{a : \{a : g_1\} \cup b : \{a : g_1\} \cup c : g_2\}$$

$$g_1 = \{d : \{\}\}$$

$$g_2 = \{c : g_2\}$$

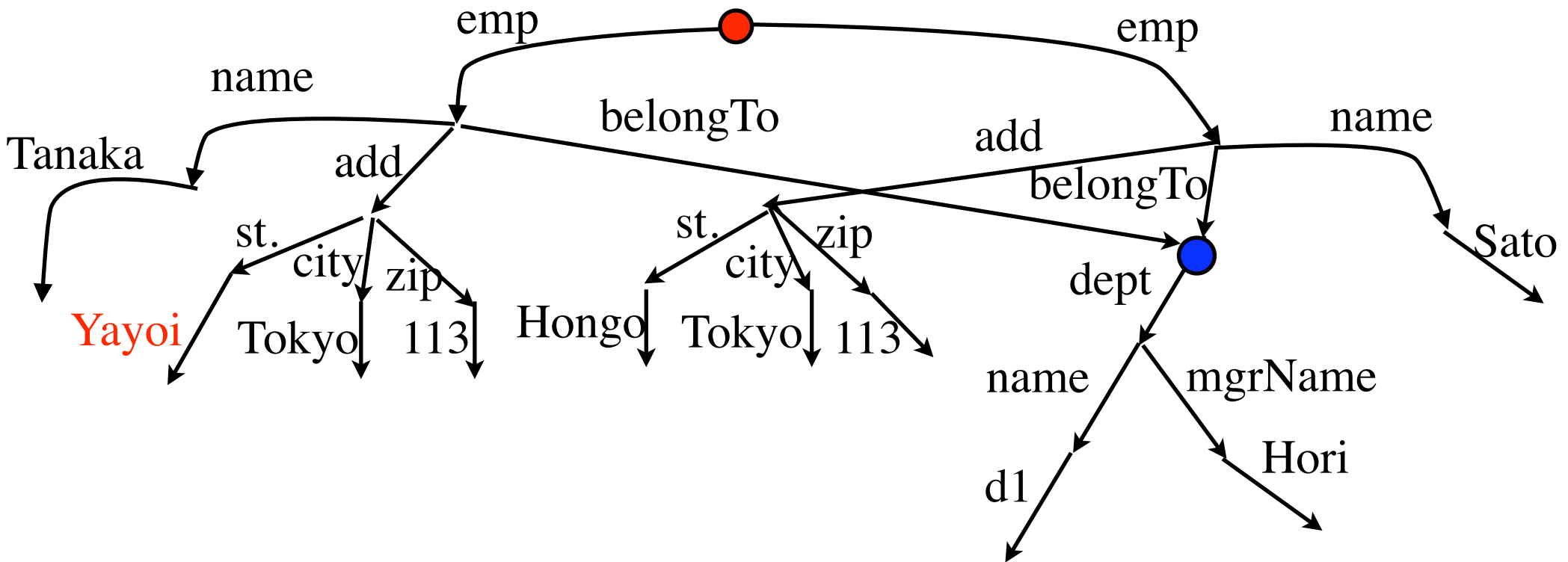


Our Graph Model



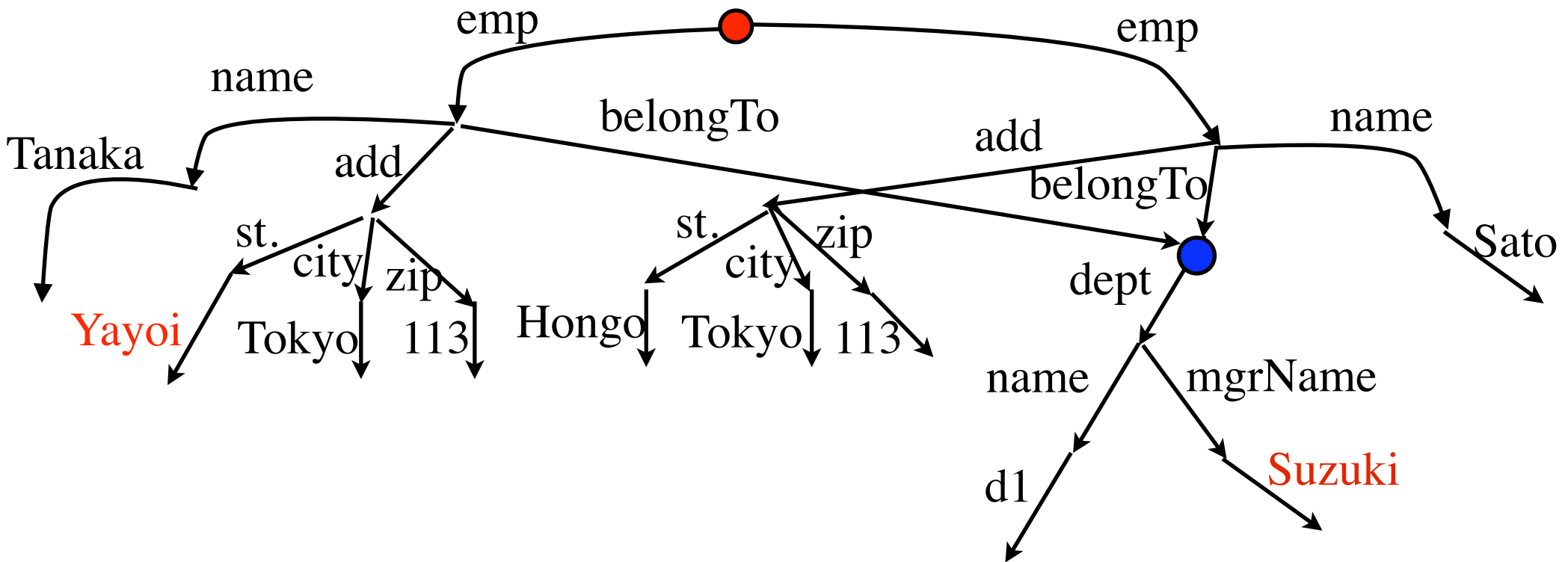
Our Graph Model

- “Tanaka” has moved to new address.



Our Graph Model

- The manager of d1 has changed.



Structural Recursion: Manipulating Graphs

Structural Recursion:

$$f(\{\}) = \{\}$$

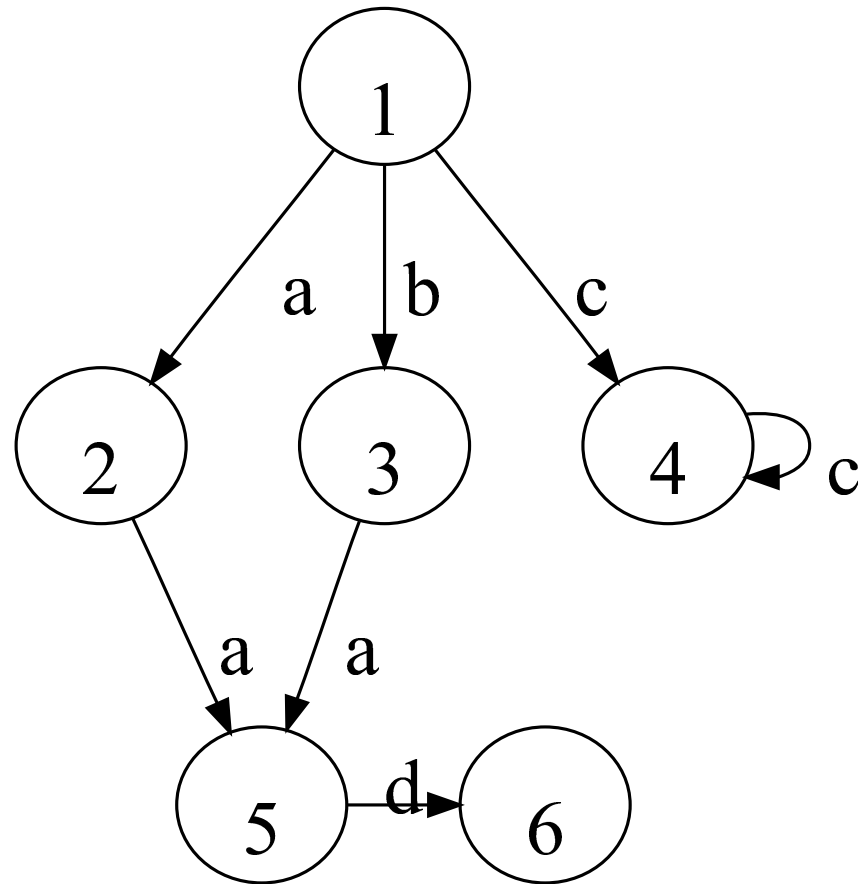
$$f(\{ \$l : \$g \}) = e \odot f(\$g)$$

$$f(g_1 \cup g_2) = f(g_1) \cup f(g_2)$$

Or written as:

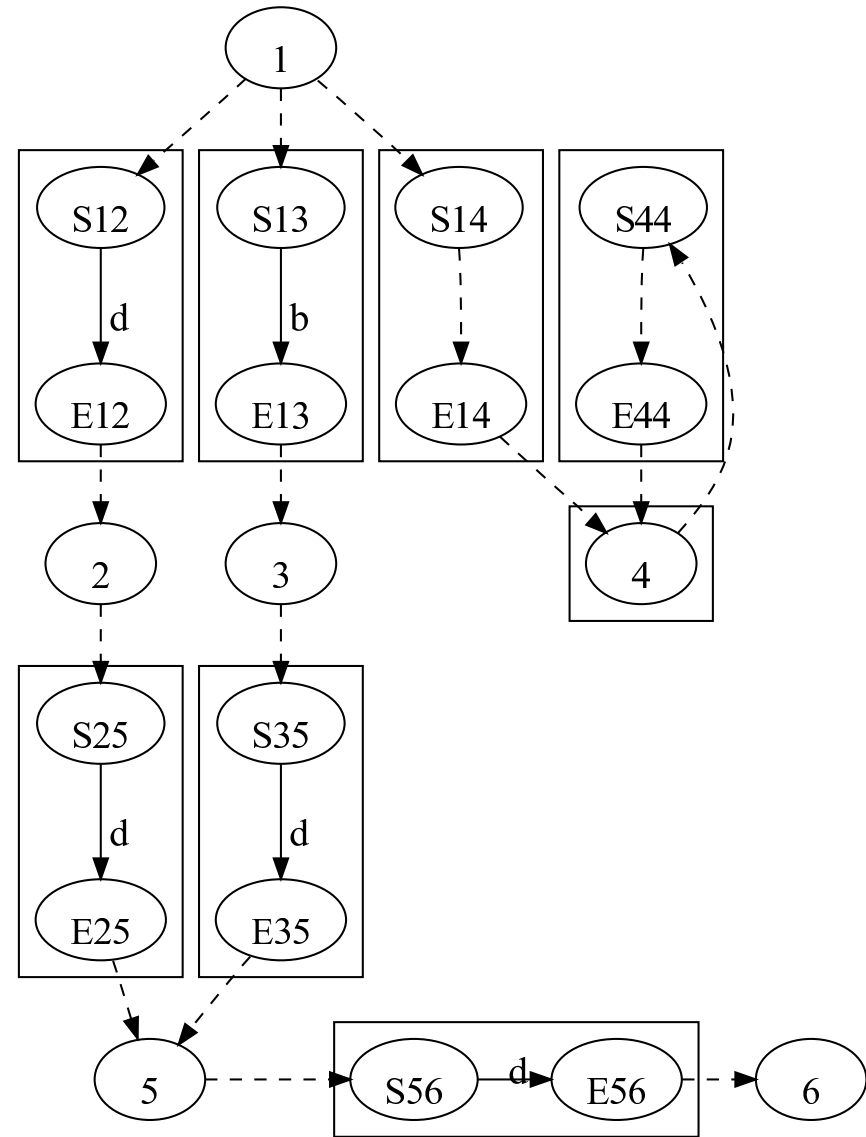
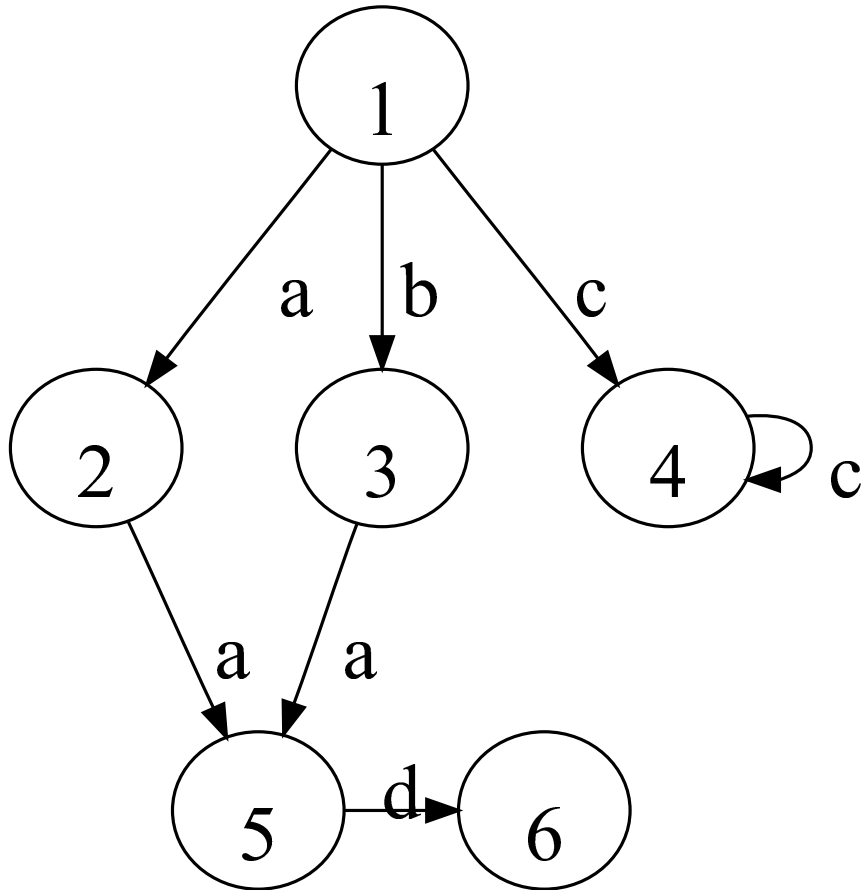
$$\text{sfun } f(\{ \$l : \$g \}) = e \odot f(\$g)$$

Structural Recursion: An Example

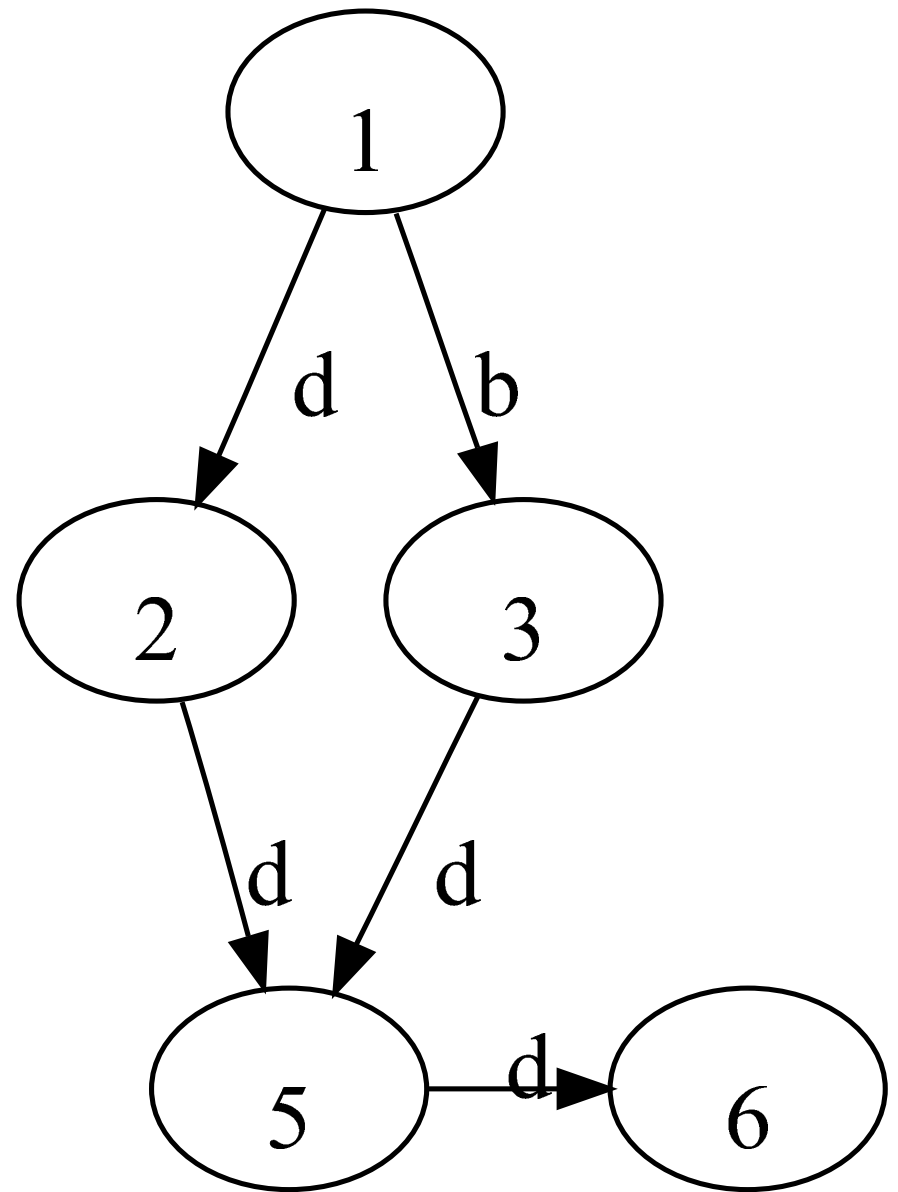
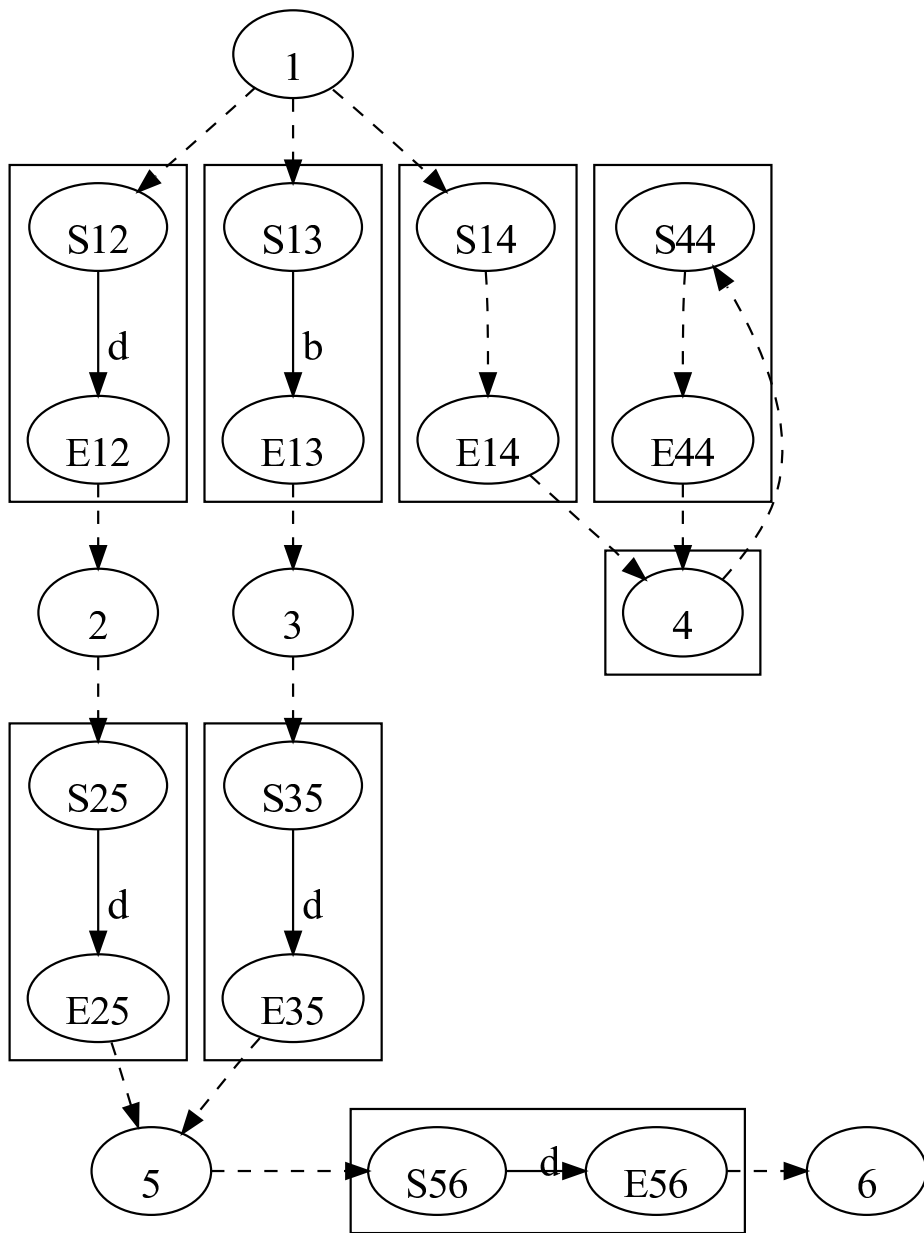


```
sfun a2d_xc ({ $l : $g }) = if $l = a then { d : a2d_xc($g) }  
                           else if $l = c then a2d_xc($g)  
                           else { $l : a2d_xc($g) }
```

The Bulk Semantics of Structural Recursion



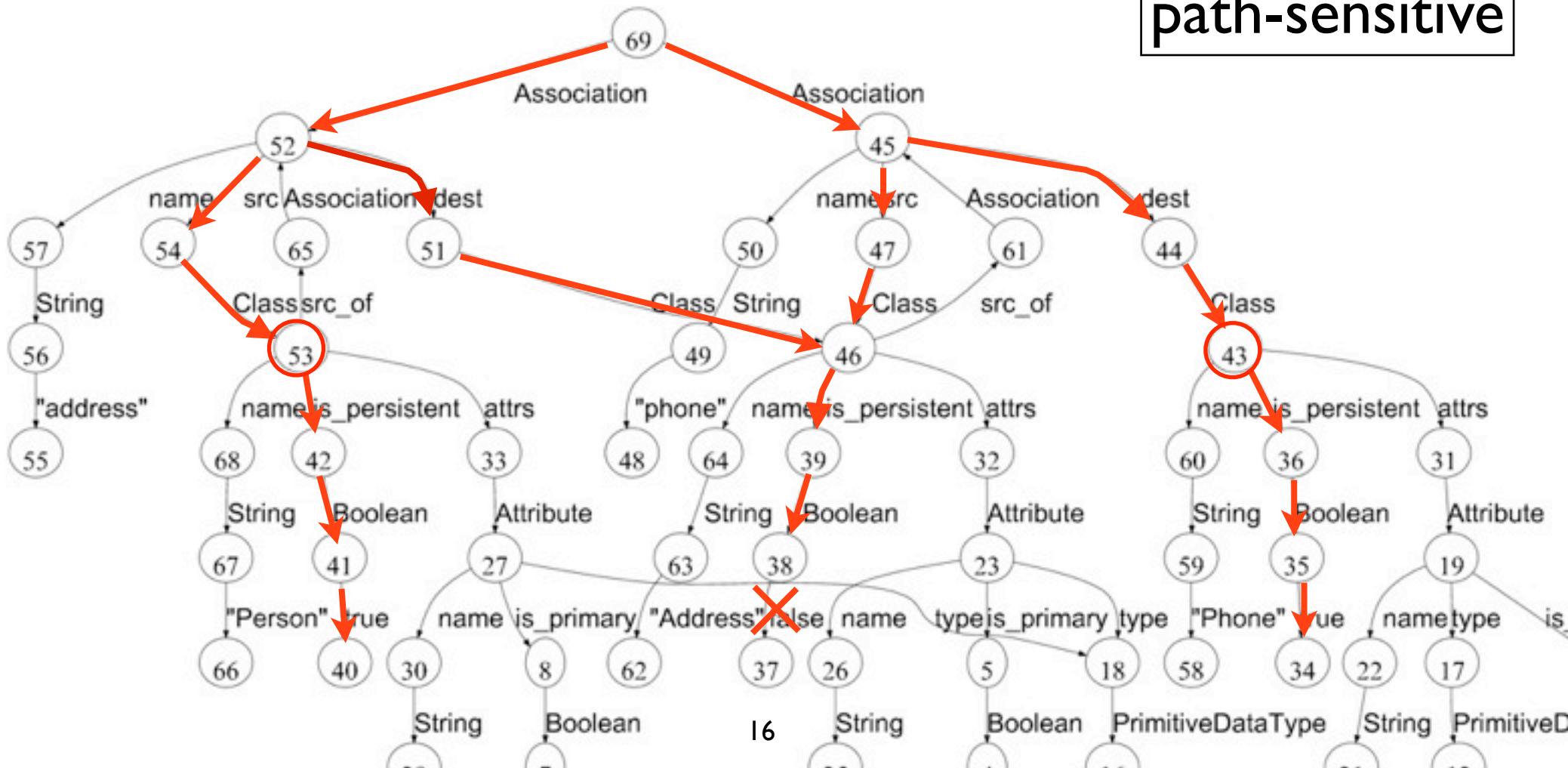
sfun $a2d_xc(\{ \$l : \$g \}) =$ if $\$l = a$ then $\{d : a2d_xc(\$g)\}$
 else if $\$l = c$ then $a2d_xc(\$g)$
 else $\{ \$l : a2d_xc(\$g) \}$



Graph Querying in UnQL

`select $class where`
`{Association.(src|dest).Class: $class} in $db,`
`{is_persistent: {Boolean : true}} in $class`

path-sensitive



UnQL⁺: An Extension of UnQL

delete ... where

extend ... with ... where

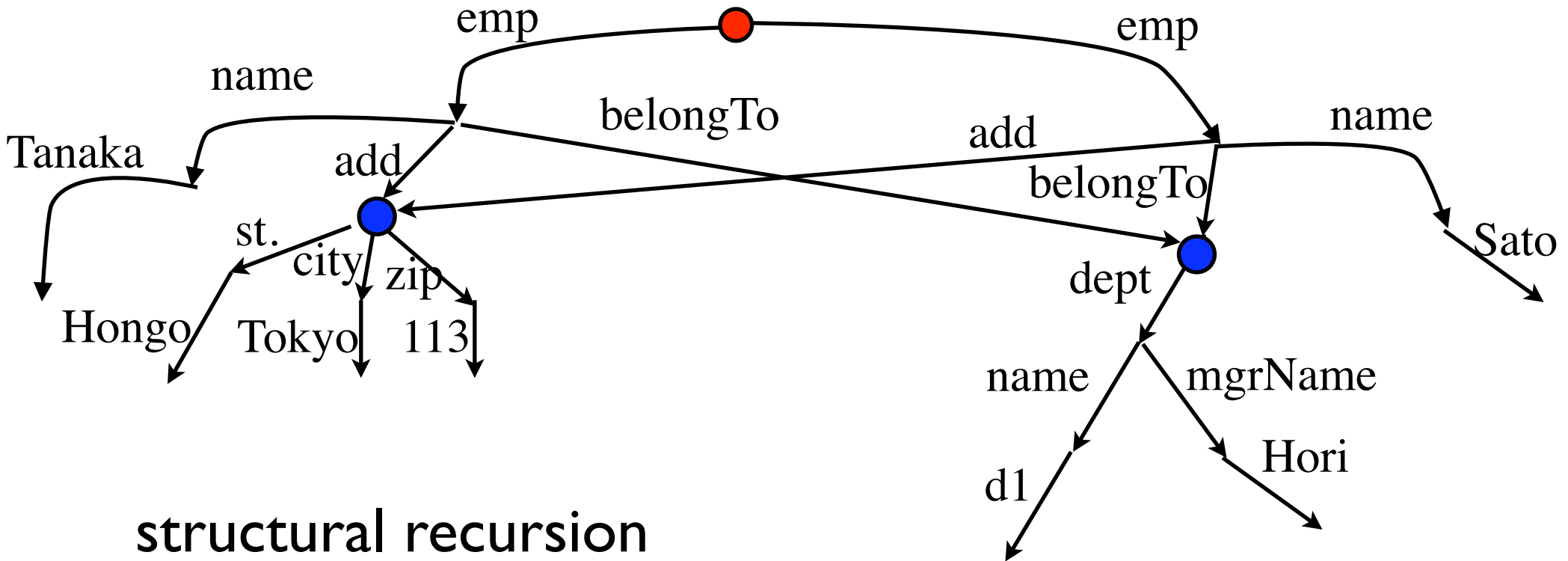
replace ... by ... where

These can be translated into **structural recursion**.

- “Tanaka” has moved to new address.

```

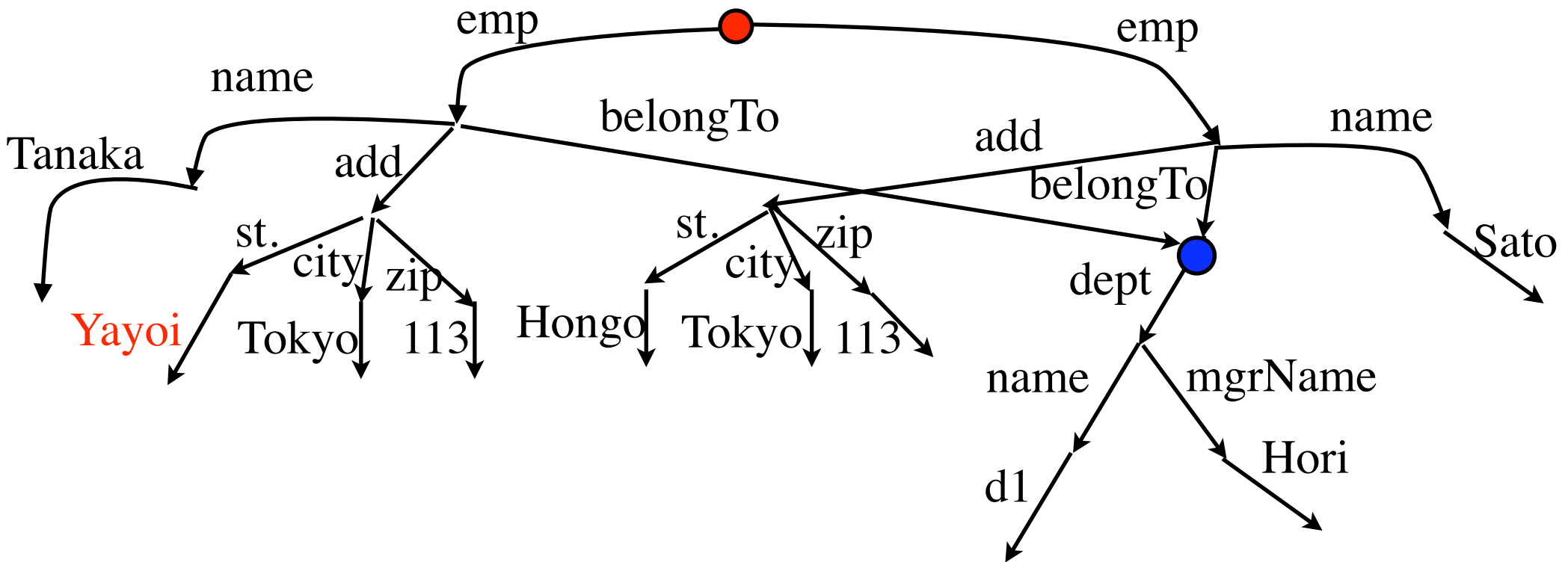
replace $tgt by {st: {Yayoi: {}}, city: {Tokyo: {}}, zip: {113: {}}}
where {emp:$e} in $db,
        {name: {Tanaka: {}}} in $e,
        {add: $tgt} in $e
  
```



- “Tanaka” has moved to new address.

```

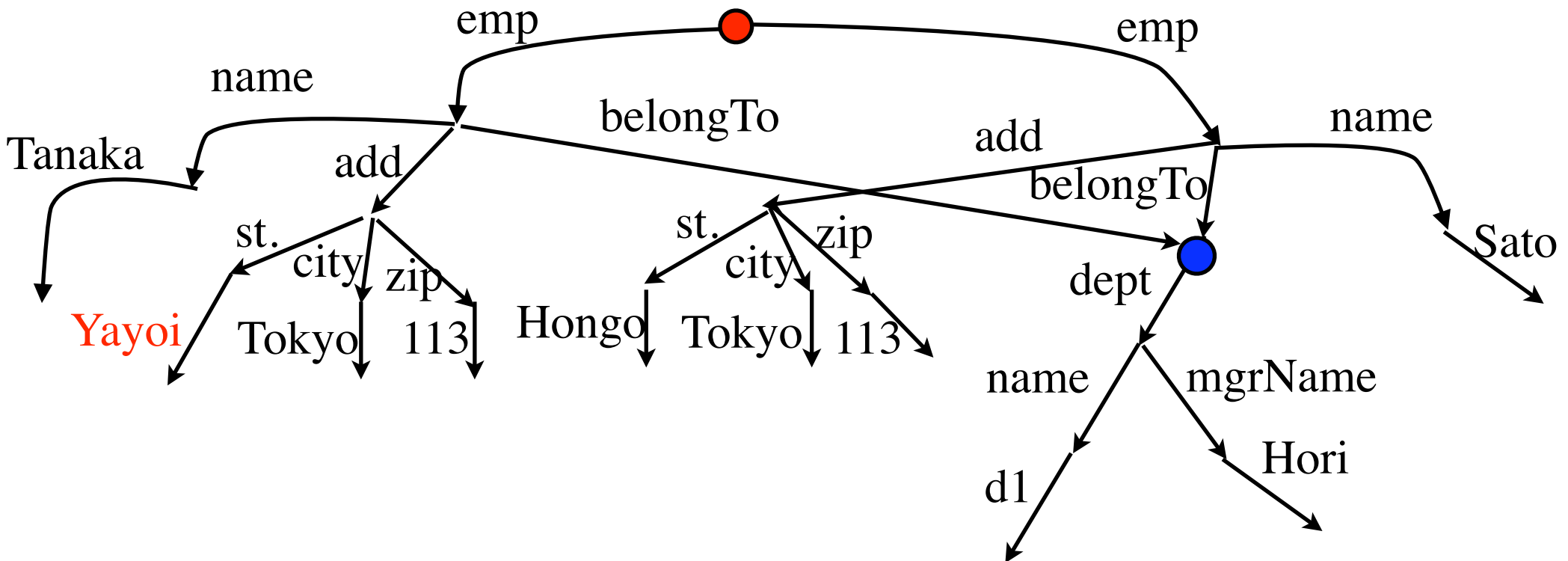
replace $tgt by {st: {Yayoi: {}}, city: {Tokyo:{}}, zip: {113: {}}}
where {emp:$e} in $db,
        {name: {Tanaka: {}}} in $e,
        {add: $tgt} in $e
  
```



- The manager of d1 has changed.

```

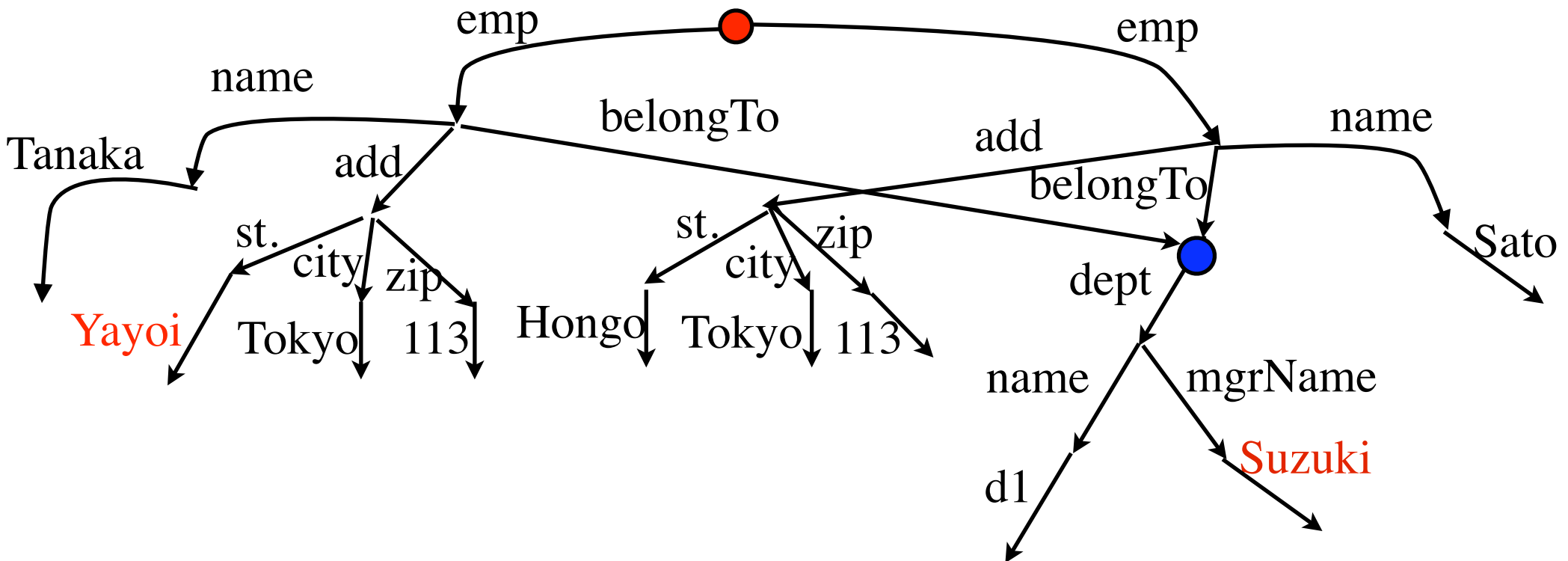
replace $tgt by {Suzuki: {}}
where {emp.belongTo.dept:$d} in $db,
        {name: {d1: {}}} in $d,
        {mgrName: $tgt} in $d
  
```



- The manager of d1 has changed.

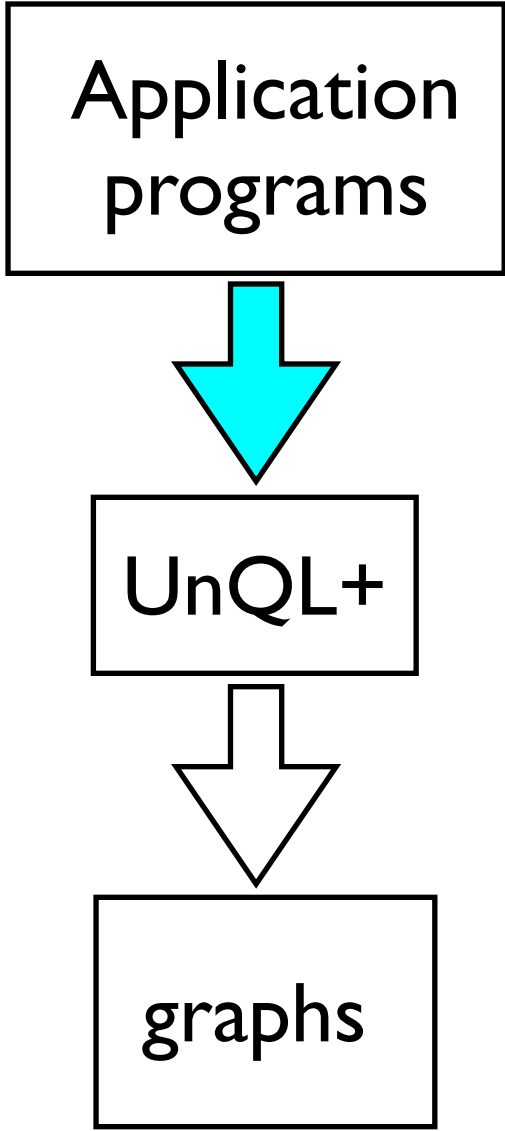
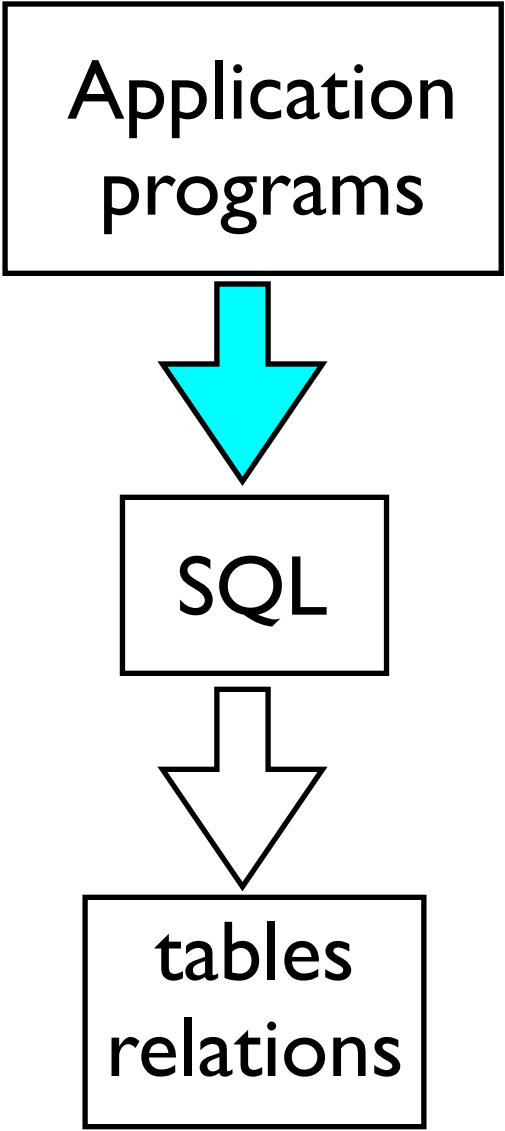
```

replace $tgt by {Suzuki: {}}
where {emp.belongTo.dept:$d} in $db,
        {name: {d1: {}}} in $d,
        {mgrName: $tgt} in $d
  
```



Conclusion

- There are two semantics in updating graphs.
- UnQL+(**replace** expression) can handle these two.



Thank you