

Algorithms From Yaner & Goel, “Visual Analogy: Viewing Analogical Retrieval and Mapping as Constraint Satisfaction Problems”

function INITDOMAINS returns Initial Target Domains	
Input:	target image and relational description
Output:	initial target element domains
<pre> 1: Let <i>Nodes</i> be a list of all the nodes in the target 2: <i>InitDomain</i>[*] $\leftarrow \emptyset$ 3: for all $w \in Nodes$ do 4: Let <i>Termss</i> be a list of all the terms in which w appears 5: <i>MappedNodes</i> \leftarrow none 6: for all $term \in Terms$ do 7: Let <i>Candidates</i> be a list of all nodes from memory incident on a term whose label matches $term$ (either incoming or outgoing as appropriate) 8: if <i>MappedNodes</i> = none then 9: <i>MappedNodes</i> $\leftarrow Candidates$ 10: else 11: <i>MappedNodes</i> $\leftarrow Candidates \cap MappedNodes$ 12: end if 13: end for 14: <i>InitDomain</i>[w] $\leftarrow MappedNodes$ 15: end for 16: return <i>InitDomain</i>[*] </pre>	

Table 1: Algorithm: Initialize Target Domains

function REDUCEDOMAINS returns Reduced Target Domains	
Input:	list of target elements, initial target domains
Output:	reduced target domains
<pre> 1: <i>InitDomain</i>[*] ← INITDOMAINS() 2: Compute the document ids for each list in <i>InitDomains</i>[*] 3: Let <i>ReferenceList</i> be the intersection across all of these lists 4: <i>Domain</i>[*] ← ∅ 5: for all $w \in Nodes$ do 6: <i>CurrentList</i> ← ∅ 7: for all $i \in InitDomain[w]$ do 8: if the document id of i is in <i>ReferenceList</i> then 9: Add i to <i>CurrentList</i> 10: end if 11: end for 12: <i>Domain</i>[w] ← <i>CurrentList</i> 13: end for 14: return <i>Domain</i>[*] </pre>	

Table 2: Algorithm: Reduce Target Domains

function FINDMATCHINGS returns Mappings from sources to target	
Input:	target description, reduced target domains, all elements in target image
Output:	list of all mappings from sources to target
<pre> 1: $n \leftarrow Nodes$ 2: Let <i>Mappings</i>[*] be nil 3: $k \leftarrow 0$ 4: <i>Open</i> ← {(nil, nil)} 5: while <i>Open</i> ≠ ∅ do 6: ($w, current$) ← POP(<i>Open</i>) 7: if $w = nil$ then 8: $w \leftarrow 1$ 9: else 10: $w \leftarrow w + 1$ 11: end if 12: for all $j \in Domain[w]$ do 13: if CONSISTENT($j, current$) then 14: $new \leftarrow APPEND(current, j)$ 15: if $w = n$ then 16: <i>Mappings</i>[k] ← new 17: $k \leftarrow k + 1$ 18: else 19: PUSH($(w, new), Open$) 20: end if 21: end if 22: end for 23: end while 24: Each item (list) in <i>Mappings</i> now corresponds to a matching from the target to some document in memory. </pre>	

Table 3: Algorithm: Find Matchings

function CONSISTENT(w, j) returns TRUE or FALSE	
Input:	candidate map (source element), current partial mapping, target description
Output:	true, if consistent, false otherwise
<pre> 1: if <i>current</i> = nil then 2: return TRUE 3: end if 4: for all $i \in \{1, \dots, w - 1\}$ do 5: if Not all relations between i and w can be found among the rela- tions between <i>current</i>[i] and j then 6: return FALSE 7: end if 8: end for </pre>	

Table 4: Algorithm: Determine if a proposed element mapping is consistent with the previous mappings