Solutions

A rete is a network (really a directed acyclic graph) used to increase the efficiency of a forward chaining system. The key idea: Variable bindings that match rule antecedents are saved and reused, rather than recomputed.

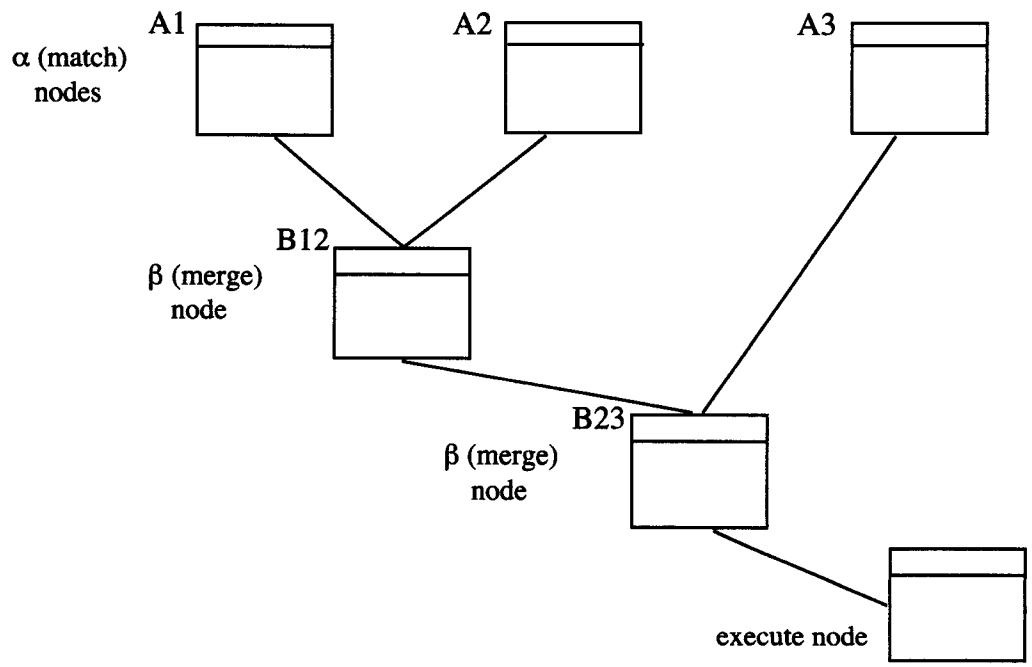
Recall that we talked about forward chaining as a procedure that loops through a set of rules, searching a database of assertions for variable bindings that match rule antecedents. When matches are found, rules are triggered and fired, producing new assertions (or removing assertions). When new assertions are added, the loop through a set of rules repeats.

The rete procedure takes advantage of two observations: (1) not many assertions are added or changed when a rule fires, and the changed assertions do not affect many rules; and (2) many rules share antecedents (i.e. their "if" components look similar). Its disadvantages: it takes up a lot of space, and if your system does a lot of removing of assertions, effort must be spent removing match information.

Each node in a rete graph represents the set of variable bindings that match an assertion or conjunction of assertions. The rete procedure works by moving assertions through this graph, saving incremental match information as it goes. A path through the graph to a leaf node represents the bindings that match the antecedents in a rule. You can think of the root node as a node that connects subgraphs for all antecedents of a particular length. (Note: Relational databases are often used to implement rete graphs: Each node, except the root node, is a table representing a relation, and each row of the table records one assertion. The match procedure uses a database Select operation to define the match, then a Project operation to find specific matches in the set of all assertions. The merge is a database Join operation, and the execute procedure does a Project operation to get the subset of matching variables needed for the consequent.)

To build a rete:
. For each antecedent, create an "alpha" node, aka a match node.
. Join a first antecedent and second antecedent to create a "beta" node, aka a merge node.
. Join each subsequent antecedent with the previous merge node to create a new merge node.
. For each consequent, create a terminal node, aka execute node, that carries out the consequent.

For each assertion:
. Add the assertion's variable bindings (or the assertion) to the appropriate match and merge nodes.
. If execute node is reached, carry out consequent(s). If consequent adds an assertion, make sure assertion is new.



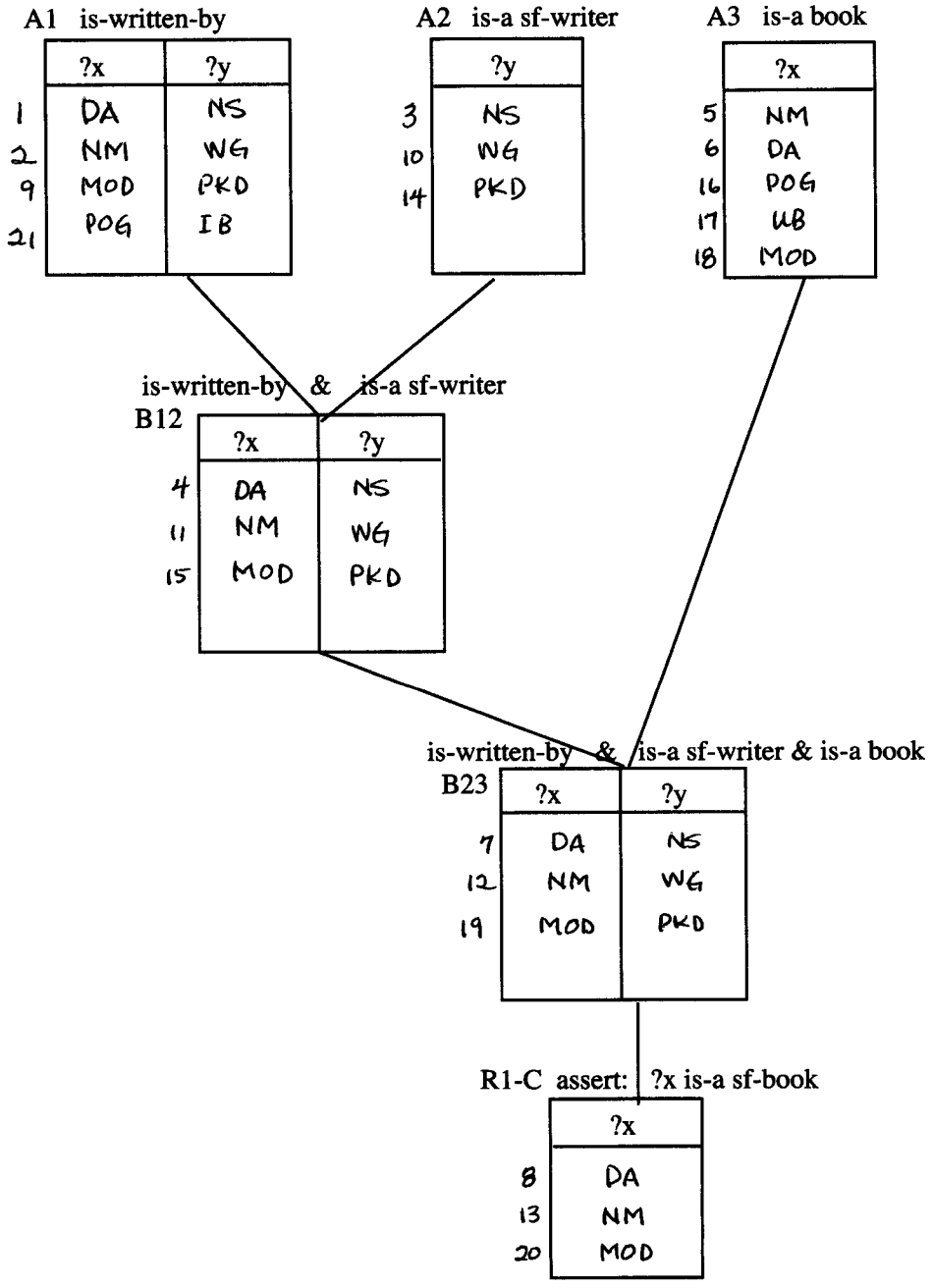
Rete Example 1

R1:

if (?x is-written-by ?y)	A1
(?y is-a science-fiction-writer)	A2
(?x is-a book)	A3
then (?x is-a science-fiction-book)	R1-C

Assertions:

TheDiamondAge is-written-by NealStephenson.
Neuromancer is-written-by WilliamGibson.
NealStephenson is-a science-fiction-writer.
Neuromancer is-a book.
TheDiamondAge is-a book.
MazeOfDeath is-written-by PhilipKDick.
WilliamGibson is-a science-fiction-writer
PhilipKDick is-a science-fiction-writer.
PlayerOfGames is-a book.
Ubik is-a book.
MazeOfDeath is-a book.
PlayerOfGames is-written-by IainBanks.



Rete Example 2: sharing structure

R1 from example 1:

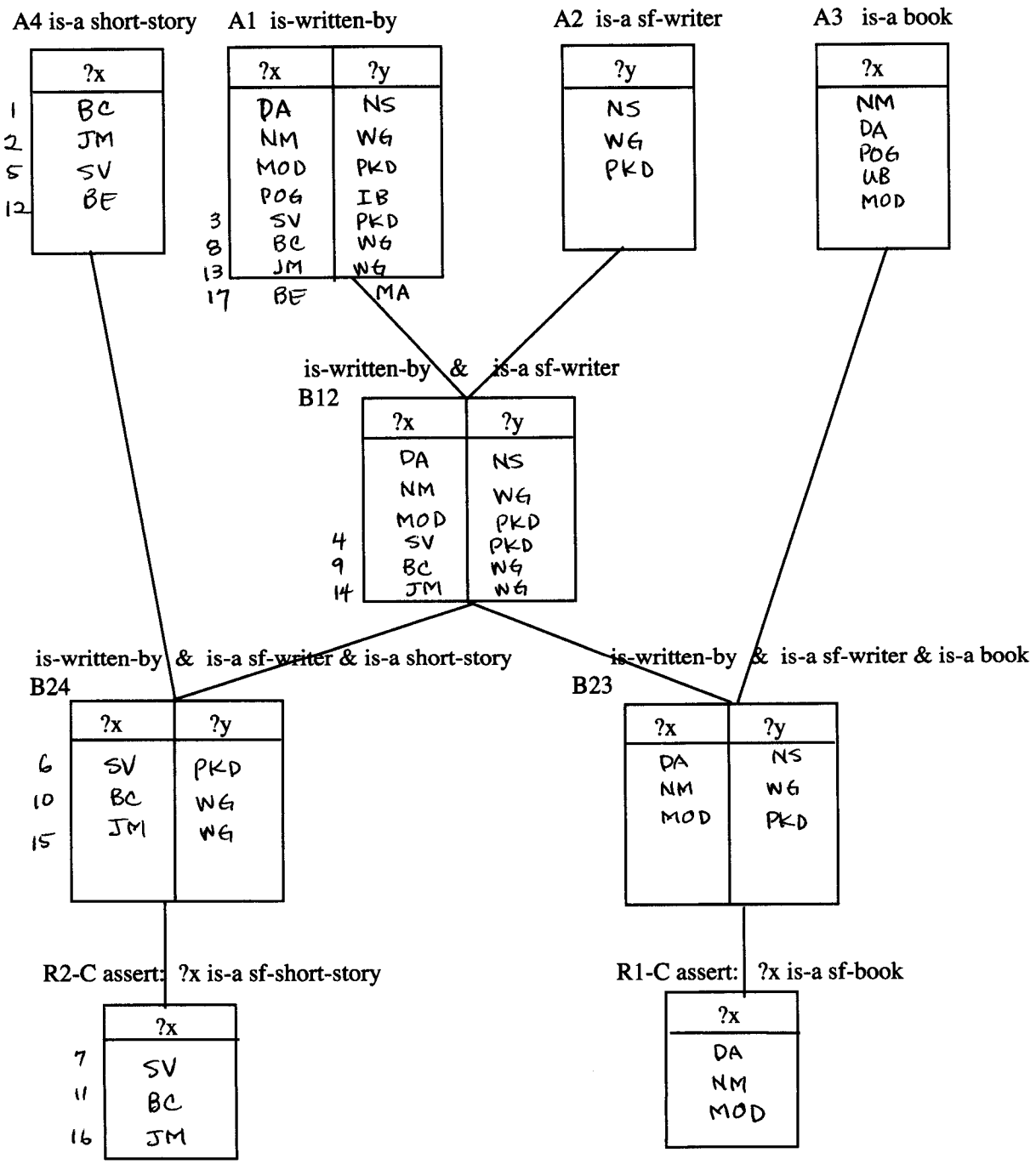
if (?x is-written-by ?y)	A1
(?y is-a science-fiction-writer)	A2
(?x is-a book)	A3
then (?x is-a science-fiction-book)	R1-C

R2:

if (?x is-written-by ?y)	A1
(?y is-a science-fiction-writer)	A2
(?x is-a short-story)	A4
then (?x is-a science-fiction-short-story)	R2-C

Assertions from example 1, plus these new assertions:

- BurningChrome is-a short-story.
- JohnnyMnemonic is-a short-story.
- SecondVariety is-written-by PhilipKDick.
- SecondVariety is-a short-story.
- BurningChrome is-written-by WilliamGibson.
- BluebeardsEgg is-a short-story.
- JohnnyMnemonic is-written-by WilliamGibson.
- BluebeardsEgg is-written-by MargaretAtwood.



Rete Example 3: sharing structure and adding an assertion

R1 from example 1:

if (?x is-written-by ?y)	A1
(?y is-a science-fiction-writer)	A2
(?x is-a book)	A3
then (?x is-a science-fiction-book)	R1-C

R2 from example 2:

if (?x is-written-by?y)	A1
(?y is-a science-fiction-writer)	A2
(?x is-a short-story)	A4
then (?x is-a science-fiction-short-story)	R2-C

R3:

if (?x is-a-book)	A3
(?x is science-fiction)	A5
then (?x is-a-science-fiction-book)	R3-C (same as R1-C)

Same assertions from example 2, plus these new ones:
Ubik is science-fiction.
PlayerOfGames is science-fiction.
Neuromancer is science-fiction.