

# ALARM \*

This system involves three people: Peter, Jane and John. Jane and John are neighbors of Peter. Peter's house has an alarm that is sensitive to earthquakes. Peter is at work, while Jane and John are at home. We want to know who calls Peter if his alarm goes off.

## 1 Rules

- A1 if (earthquake)  
then Peter alarm on
- A2 if (?x alarm on)  
(neighbor ?y ?x )  
then (?y hears alarm-of ?x )
- A3 if (neighbor ?x ?y )  
then (neighbor ?y ?x )
- A4 if (earthquake)  
then (radio-announcement)
- A5 if (radio-announcement)  
(listens-to-radio ?x )  
then (hears-announcement ?x )
- A6 if (does-not-listen-to-radio ?x )  
then (does-not-hear-announcement ?x )
- A7 if (?x hears alarm-of ?y )  
(does-not-hear-announcement ?x )  
then (call ?x ?y )

## 2 Initial working memory

(neighbor Jane Peter)  
(neighbor Peter John)  
(earthquake)  
(listens-to-radio Jane)  
(does-not-listen-to-radio John)

---

\*Based on Alarm network from *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference (Revised Second Printing)* by Judea Pearl.

### 3 Forward Chaining

Run the forward chainer to deduce who calls Peter using the set of assertion in the working memory given above. At each step, write down the triggered rules and any rule instance bindings (if any). Assume that *all triggered rules fire* and new assertions are added after existing ones. Terminate when no further assertions can be made. You can abbreviate clauses as long as there are no ambiguity.

	Triggered rule instances	New
①	1 3: $x = \text{Jane}$ $y = \text{Peter}$ 3: $x = \text{Peter}$ $y = \text{John}$ 4 6: $x = \text{John}$	(Peter alarm on) (neighbor Peter Jane) (neighbor John Peter) (radio-ann.) (does-not-hear-radio-ann. John)
②	2: $x = \text{Peter}$ $y = \text{Jane}$ 2: $x = \text{Peter}$ $y = \text{John}$ 5: $x = \text{Jane}$	(Jane hears-alarm Peter) (John " " ) (hears-ann. Jane)
③	7: $x = \text{John}$ , $y = \text{Peter}$	(call John Peter)

## 4 Backward Chaining

Someone asked you whether Jane called Peter so you run a backward chainer to check whether or not you can reach that conclusion.

You use the same assertions as in your forward chaining system. You also use the same 7 rules plus an additional one that you add to the top of the list:

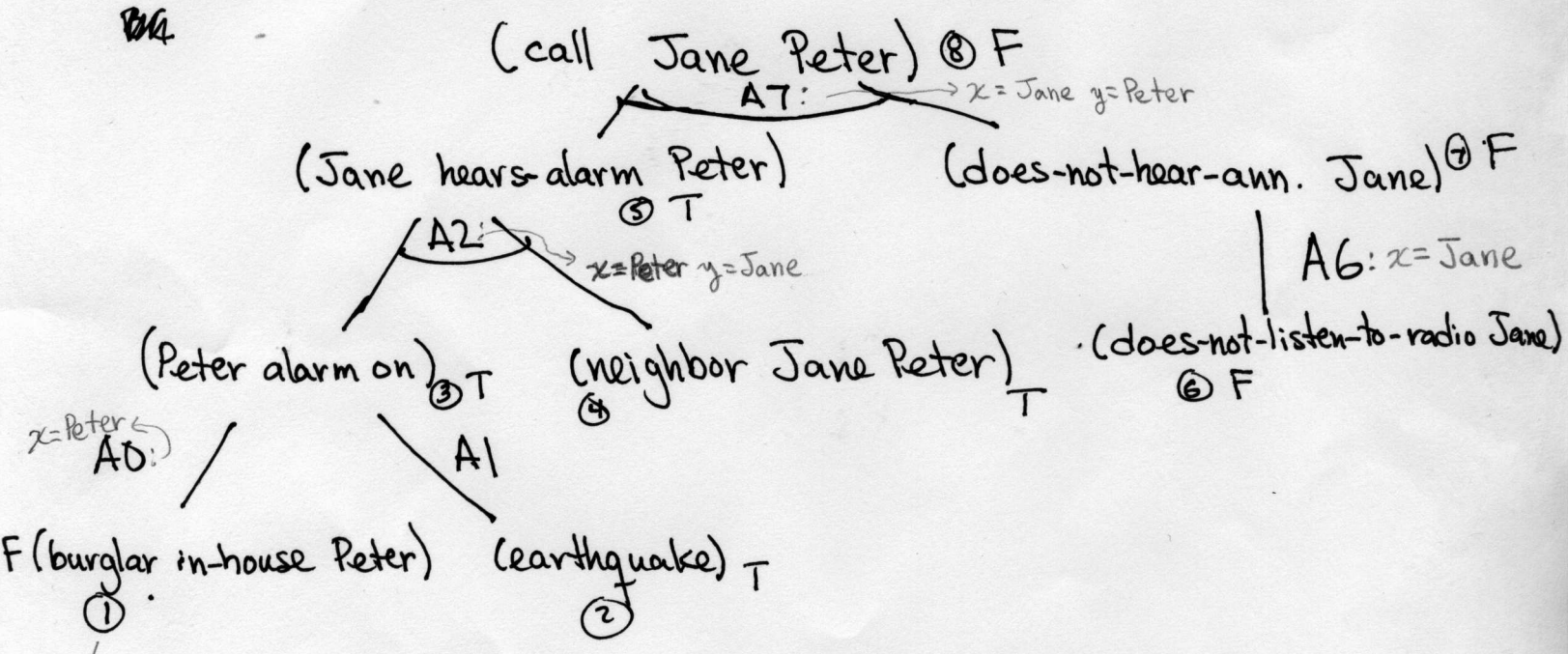
A0 if (burglar in-house-of ?x)  
then (?x alarm on)

You then ask your backward chainer to prove the following statement:  
(call Jane Peter)

Using this assertion as the root node, draw the goal (and/or) tree that your system uses to prove the assertion. Assume that your system uses rule ordering as a conflict resolution strategy. Also assume that if an assertion cannot be proven via rules or existing assertions, that it fails. (In other words, your system does not query for an answer.) Label each branch of the tree with the name of the rule (e.g. A1) that it represents.

*or disprove*

*BA*



→ #'s correspond to order of when "hypothesis" is shown true or false.