Inference in rule-based systems

Inferencing

- The inference engine uses one of several available forms of inferencing.

- By inferencing we mean the method used in a knowledge-based system (KBS) to process the supplied data, and the stored knowledge, so as to produce correct conclusions.

Distinctive features of KBS (1)

- **The need for heuristic reasoning.**

  - ☐ Conventional computer programs are built around algorithms: reasoning strategies which are guaranteed to find the solution to whatever the problem is, if there is such a solution.

  - ☐ For the large, difficult problems with which KBS are frequently concerned, it may be necessary to employ heuristics: strategies that often lead to the correct solution, but which also sometimes fail.

Distinctive features of KBS (2)

☐ Humans use heuristics a great deal in their problem solving. Of course, if the heuristic does fail, it is necessary for the problem solver to either pick another heuristic, or know that it is appropriate to give up.

☐ The rules, found in the knowledge bases of rule-based systems, are very often heuristics.

Distinctive features of KBS (3)

- **Large solution spaces.**
  - □ One way to treat a problem is to describe the system concerned as a state space –
    - ■ a collection of states that it can get into, with a description of the transitions that take you from one state to another.
    - ■ Some of these states are solutions to the problem.
  - □ In this approach, the way to solve a problem is to search in a systematic way until you have found a path from the start state to a goal state.

Distinctive features of KBS (4)

- **Large solution spaces.**

  - In scheduling and design problems, there may be many millions of possible solutions to the problem as presented.

  - It is not possible to consider each one in turn, to find the right (or best) solution; heuristically-guided search is required.

Distinctive features of KBS (5)

- **Multiple solutions.**
  - ☐ In planning or design tasks, a single solution will probably be enough.
  - ☐ In diagnostic tasks, all possible solutions are probably needed.

Distinctive features of KBS (6)

- **Reasoning with uncertainty.**
  - ☐ Rules in the knowledge base may only express a probability that a conclusion follows from certain premises, rather than a certainty.
  - ☐ This is particularly true of medicine and other life sciences.
  - ☐ The items in the knowledge base must reflect this uncertainty, and the inference engine must process the uncertainties to give conclusions that are accompanied by a likelihood that they are true or correct.

Inference in rule-based systems

- Two control strategies: forward chaining and backward chaining

Inference in rule-based systems

- Forward chaining: working from the facts to a conclusion. Sometimes called the data-driven approach. To chain forward, match data in working memory against 'conditions' of rules in the rule-base.

  - ☐ To chain forward, match data in working memory against 'conditions' of rules in the rule-base.

  - ☐ When one of them fires, this is liable to produce more data.

  - ☐ So the cycle continues

Inference in rule-based systems

- Backward chaining: working from the conclusion to the facts. Sometimes called the goal-driven approach.

  - To chain backward, match a goal in working memory against 'conclusions' of rules in the rule-base.

  - When one of them fires, this is liable to produce more goals.

  - So the cycle continues.

Forward & backward chaining

- e.g. Here are two rules:

If corn is grown on poor soil, then it will rot.

If soil hasn't enough nitrogen, then it is poor soil.

- Forward chaining: This soil is low in nitrogen; therefore this is poor soil; therefore corn grown on it will rot.

- Backward chaining: This corn is rotten; therefore it must have been grown on poor soil; therefore the soil must be low in nitrogen.

Forward chaining

- More realistically,
  - the forward chaining reasoning would be: there's something wrong with this corn. So I test the soil. It turns out to be low in nitrogen. If that's the case, corn grown on it will rot.

Backward chaining

- More realistically,

  □ the backward chaining reasoning would be: there's something wrong with this corn. Perhaps it is rotten; if so, it must have been grown on poor soil; if so, the soil must be low in nitrogen. So test for low nitrogen content in soil, and then we'll know whether the problem is rot.

Example- A rule-based system for the classification of animals

1. IF (Animal has hair) or (Animal drinks milk) THEN Animal is a mammal

2. IF (Animal has feathers) or ((Animal can fly) and (Animal lays eggs)) THEN Animal is a bird

3. IF (Animal is a mammal) and ((Animal eats meat) or ((Animal has pointed_teeth) and (Animal has claws) and (Animal has forward_pointed_eyes))) THEN Animal is a carnivore

4. IF (Animal is a carnivore) and (Animal has tawny_colour) and (Animal has dark_spots) THEN Animal is a cheetah

5. IF (Animal is a carnivore) and (Animal has tawny_colour) and (Animal has black_stripes) THEN Animal is a tiger

6. IF (Animal is a bird) and (Animal cannot flies) and (Animal can swim) THEN Animal is a penguin

7. IF (Animal is a bird) and (Animal has large_wingspan) THEN Animal is a albatros

Example- A rule-based system for the classification of animals

- Assume there are the following facts in the working memory
  - Jimmy has hair, jimmy has pointed_teeth, jimmy has claws, jimmy has forward_pointed_eyes, jimmy has black_stripes, jimmy has tawny_colour

- Suppose that we want to see whether jimmy is a tiger using backward chaining. How will it work?

- Suppose that we want to see what jimmy is using forward chaining. How will it work?

Example- Backward chaining for the classification of animals

- Given the following facts:

  Fact (a):   Jimmy has hair,

  Fact (b):   jimmy has pointed_teeth,

  Fact (c):   jimmy has claws,

  Fact (d):   jimmy has forward_pointed_eyes,

  Fact (e):   jimmy has black_stripes,

  Fact (f):   jimmy has tawny_colour

- Backward chaining

  □ We are asking the question "Is jimmy a tiger?"

  □ First, we search for a fact which give the answer or a rule the answer could be inferred

    ■ Rule 5 if true would infer "jimmy is a tiger".

Example- Backward chaining for the classification of animals

☐ Next, we check the conditions of rule 5.

- Is "jimmy has tawny_colour" true?
  - ☐ Yes, given by fact (f).
- Is "jimmy has black_stripes" true?
  - ☐ Yes, given by fact (e).
- Is "jimmy is a carnivore" true?
  - ☐ None of the facts given can answer the question.
  - ☐ However, Rule 3 is related to it.

Example- Backward chaining for the classification of animals

- We check the conditions for Rule 3.
  - Is "jimmy eats meat" true? Or Is "jimmy has pointed_teeth and claws and forward_pointing_eyes" true?
    - Yes, given by facts (b), (c) & (d).
  - Is "jimmy is a mammal" true"?
    - None of the facts given can answer the question.
    - However, Rule 1 is related to it.
- We check the conditions for Rule 1.
  - Is "jimmy has hair or jimmy drinks milk" true?
  - Yes, jimmy has hair.
  - The fact (a) directly gives the answer.
- Hence, Rules 1, 3, 5 are used and "Jimmy is a tiger" is concluded.

Example- Forward chaining for the classification of animals

- **Forward Chaining**
  - The fact (a) matches the condition for Rule 1.
    - Jimmy has hair ➔ jimmy is a mammal
    - The conclusion "jimmy is a mammal" gives a new fact.
  - The new fact "Jimmy is a mammal" and facts (b), (c) & (d) matches the conditions for Rule 3.
    - "jimmy is a mammal" and "jimmy has pointed_teeth and claws and forward_pointing_eyes" ➔ jimmy is a carnivore.
    - Again, the conclusion "jimmy is a carnivore" is a new fact.

Example- Forward chaining for the classification of animals

- The new fact "jimmy is a carnivore" and fact (f) matches the conditions for Rule 5.
  - "jimmy is a carnivore" and "jimmy has tawny_colour and black_stripes" ➔ jimmy is a tiger.
- No other facts matches any of the rules.
- The conclusion "jimmy is a tiger" answers the question.

Forward & backward chaining

- The choice of strategy depends on the nature of the problem.

- Assume the problem is to get from facts to a goal (e.g. symptoms to a diagnosis).

Forward & backward chaining

Backward chaining is the best choice if:

- The goal is given in the problem statement, or can sensibly be guessed at the beginning of the consultation;

or:

- The system has been built so that it sometimes asks for pieces of data (e.g. "please now do the gram test on the patient's blood, and tell me the result"), rather than expecting all the facts to be presented to it.

Forward & backward chaining

Backward chaining

- This is because (especially in the medical domain) the test may be
  - expensive,
  - or unpleasant,
  - or dangerous for the human participant

so one would want to avoid doing such a test unless there was a good reason for it.

Forward & backward chaining

Forward chaining is the best choice if:

- All the facts are provided with the problem statement;

or:

- There are many possible goals, and a smaller number of patterns of data;

or:

- There isn't any sensible way to guess what the goal is at the beginning of the consultation.

Forward & backward chaining

- Note also that
  - a backward-chaining system tends to produce a sequence of questions which seems focused and logical to the user,
  - a forward-chaining system tends to produce a sequence which seems random & unconnected.

- If it is important that the system should seem to behave like a human expert, backward chaining is probably the best choice.

Forward & backward chaining

- Some systems use *mixed chaining*, where some of the rules are specifically used for chaining forwards, and others for chaining backwards. The strategy is for the system to chain in one direction, then switch to the other direction, so that:
  - the diagnosis is found with maximum efficiency;
  - the system's behaviour is perceived as "human".

Problem decomposition into an and-or graph

- A technique for reducing a problem to a production system.

- One particular form of intermediate representation.

  □ A structured representation of the knowledge, which is not yet in the form of code that can be put into KBS's knowledge base.

Problem decomposition into an and-or graph

- A technique for reducing a problem to a production system, as follows:

  - The principle goal is identified; it is split into two or more sub-goals; these, too are split up.

  - A goal is something you want to achieve. A sub-goal is a goal that must be achieved in order for the main goal to be achieved.

Problem decomposition into an and-or graph

- A graph is drawn of the goal and sub-goals.

- Each goal is written in a box, called a node, with its subgoals underneath it, joined by links.

Problem decomposition into an and-or graph

☐The leaf nodes at the bottom of the tree

- the boxes at the bottom of the graph that don't have any links below them

- are the pieces of data needed to solve the problem.
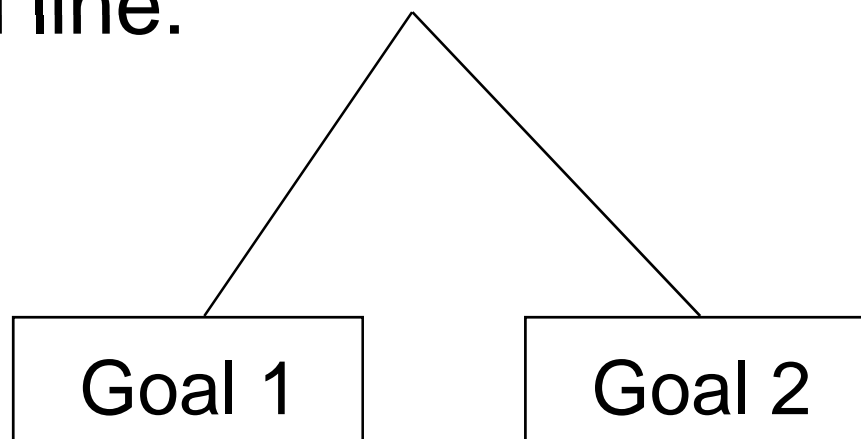
Problem decomposition into an and-or graph

- A goal may be split into 2 (or more) sub-goals, BOTH of which must be satisfied if the goal is to succeed; the links joining the goals are marked with a curved line, like this:

```
          /\
         /  )\
        /  /  \
       /        \
      /          \
 ┌────────┐   ┌────────┐
 │ Goal 1 │   │ Goal 2 │
 └────────┘   └────────┘
```
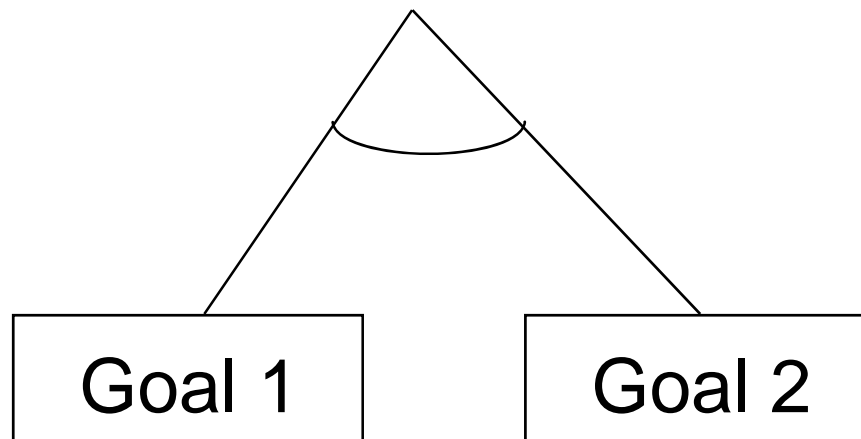
Problem decomposition into an and-or graph

- Or a goal may be split into 2 (or more) sub-goals, EITHER of which must be satisfied if the goal is to succeed; the links joining the goals aren't marked with a curved line:

```
        /\
       /  \
      /    \
  +--------+   +--------+
  | Goal 1 |   | Goal 2 |
  +--------+   +--------+
```

Problem decomposition into an and-or graph

- Example

- "The function of a financial advisor is to help the user decide whether to invest in a savings account, or the stock market, or both. The recommended investment depends on the investor's income and the current amount they have saved:

Problem decomposition into an and-or graph

☆ Individuals with inadequate savings should always increase the amount saved as their first priority, regardless of income.

🕐 Individuals with adequate savings and an adequate income should consider riskier but potentially more profitable investment in the stock market.

Problem decomposition into an and-or graph

⌚ Individuals with low income who already have adequate savings may want to consider splitting their surplus income between savings and stocks, to increase the cushion in savings while attempting to increase their income through stocks.

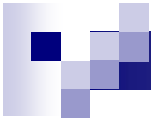# Problem decomposition into an and-or graph

⌖ The adequacy of both savings and income is determined by the number of dependants an individual must support.

There must be at least £3000 in the bank for each dependant.

An adequate income is a steady income, and it must supply at least £9000 per year, plus £2500 for each dependant."
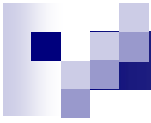
Problem decomposition into an and-or graph

- How can we turn this information into an and-or graph?

- Step 1: decide what the ultimate advice that the system should provide is.

  It's a statement along the lines of "The investment should be X", where X can be any one of several things.

■ Start to draw the graph by placing a box at the top:

```
┌─────────────────────┐
│    Advise user:     │
│    investment       │
│    should be X      │
└─────────────────────┘
```
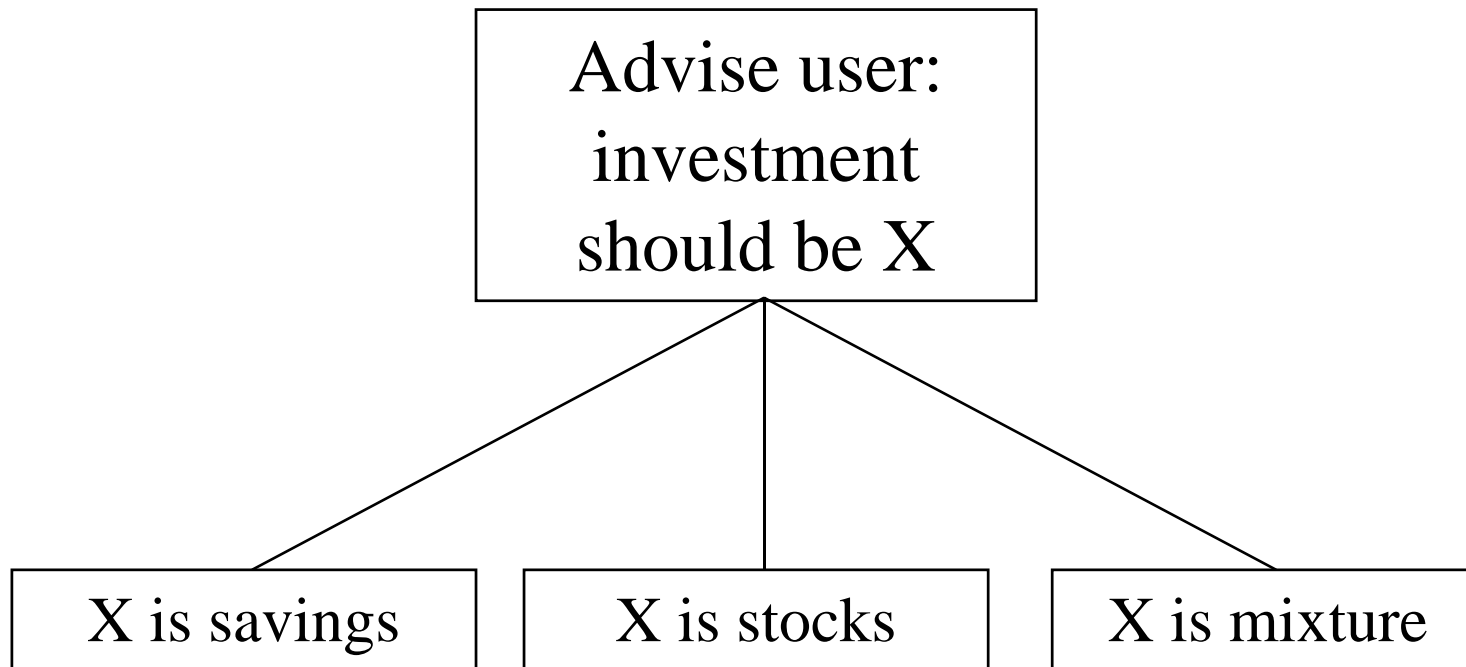
- **Step 2: decide what sub-goals this goal can be split into.**

  In this case, X can be one of three things: savings, stocks or a mixture.

  Add three sub-goals to the graph. Make sure the links indicate "or" rather than "and".

- **Steps 3a, 3b and 3c: decide what sub-goals each of the goals at the bottom of the graph can be split into.**
  - ☐ It's only true that "X is savings" if "savings are inadequate". That provides a subgoal under "X is savings"
  - ☐ It's only true that "X is stocks" if "savings are adequate" and "income is adequate. That provides two subgoals under "X is stocks" joined by "and" links.
  - ☐ Similarly, there are two subgoals under "X is mixture" joined by "and" links.
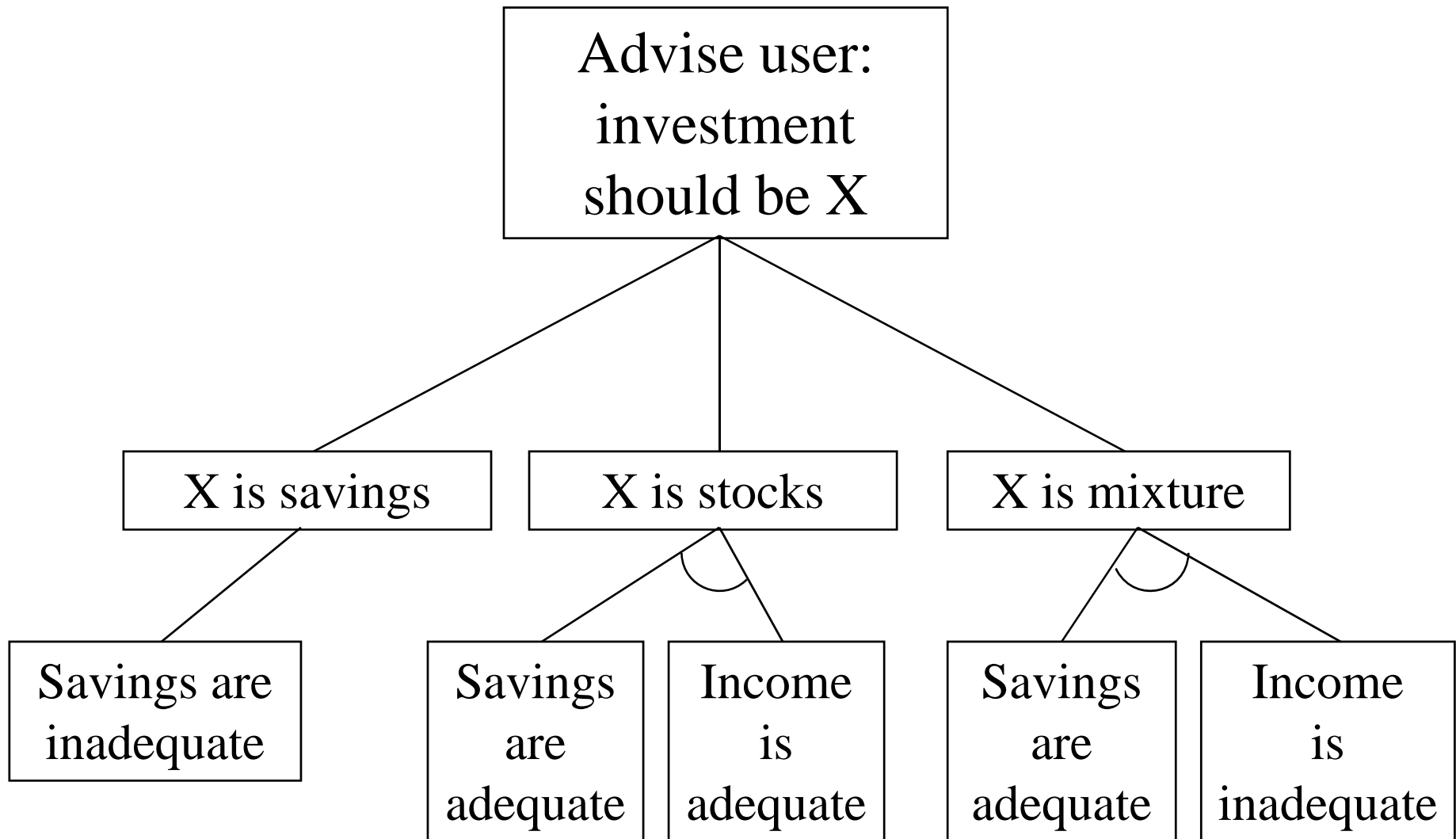
- The next steps (4a,4b,4c,4d & 4e) mainly involve deciding whether something's big enough.
  - Step 4a: savings are only inadequate if they are smaller than a certain figure (let's call it Y).
  - Step 4b: savings are only adequate if they are bigger than this figure (Y).
  - Step 4c: income is only adequate if it is bigger than a certain figure (let's call it W), and also steady.
  - Step 4d is the same as 4b. Step 4e is like 4c, but "inadequate", "smaller" and "not steady".
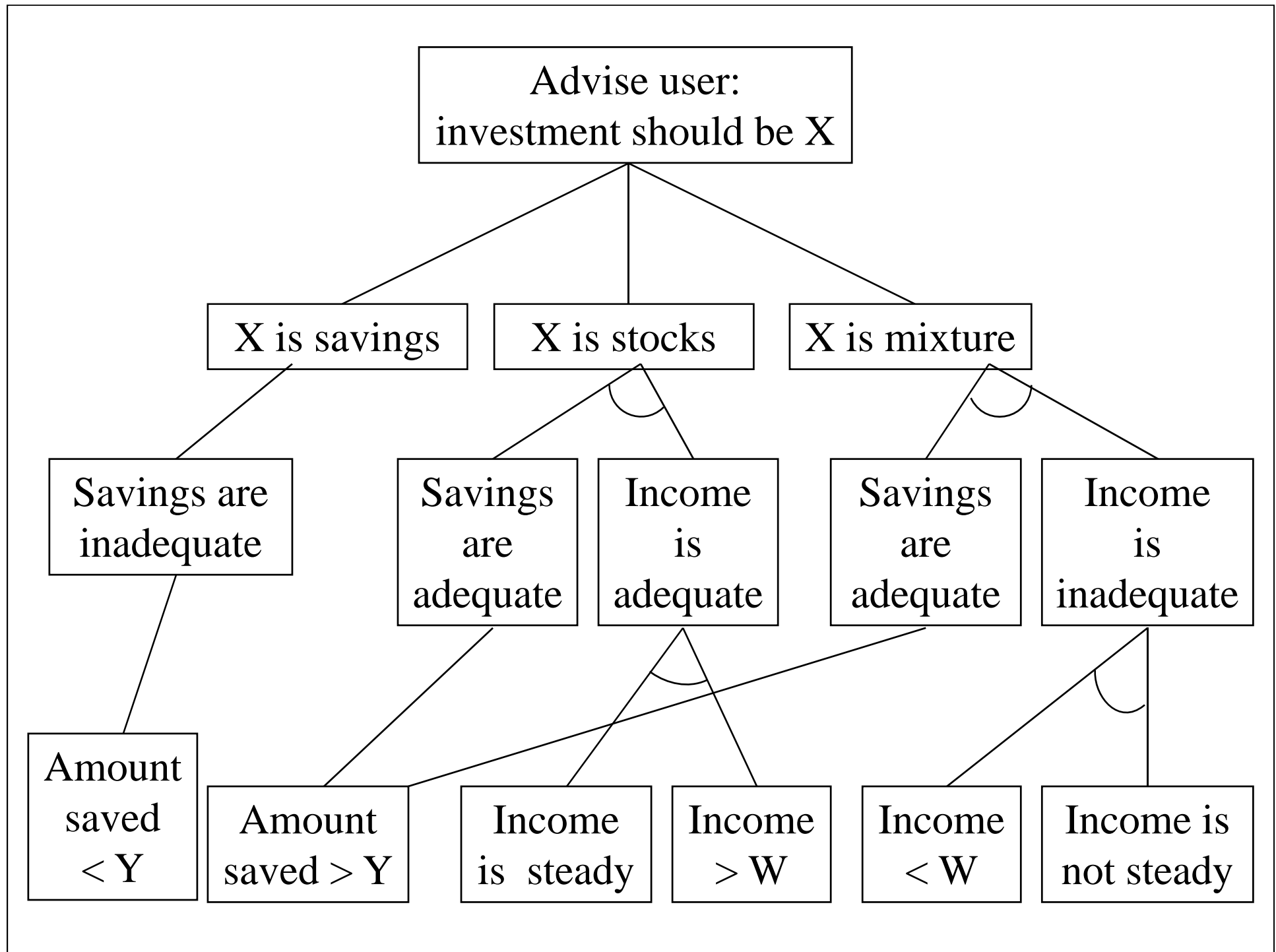
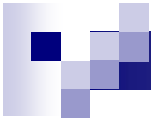- Now we need a box in which the value of Y is calculated:

Y is Z times 3000
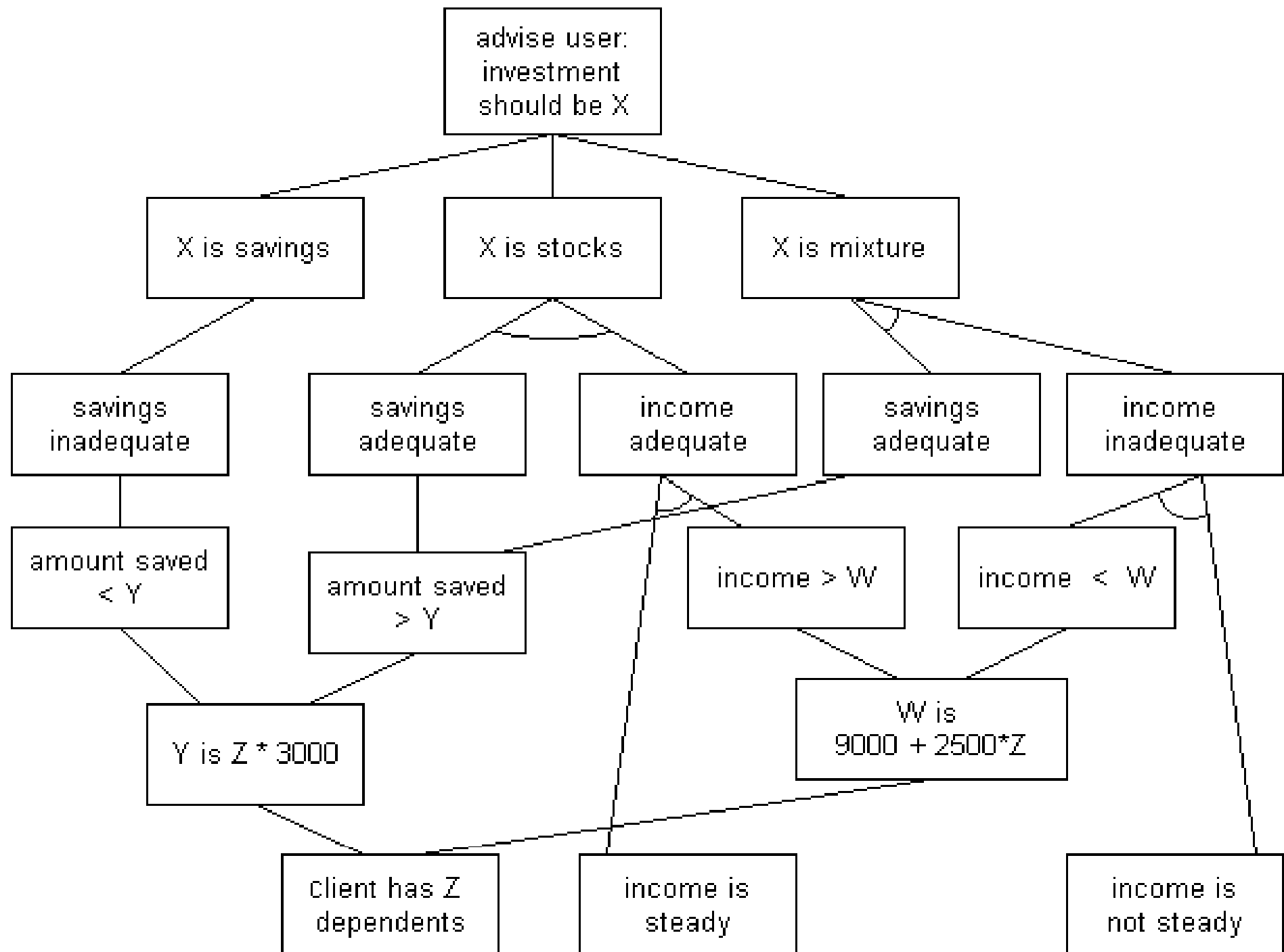
and we need a box in which the value of W is calculated:

W is 9000 plus 2500 times Z

- Z is the number of dependants, so we need a box in which this value is obtained:

Client has Z dependants

- We can now add these last three boxes into the bottom layers of the graph in the same way as we've added all the others:

```
                        ┌─────────────────┐
                        │   advise user:  │
                        │   investment    │
                        │   should be X   │
                        └─────────────────┘
              ┌──────────────────┼──────────────────┐
      ┌───────────────┐   ┌───────────────┐   ┌───────────────┐
      │  X is savings │   │  X is stocks  │   │  X is mixture │
      └───────────────┘   └───────────────┘   └───────────────┘
```

- advise user: investment should be X
- X is savings
- X is stocks
- X is mixture
- savings inadequate
- savings adequate
- income adequate
- savings adequate
- income inadequate
- amount saved < Y
- amount saved > Y
- income > W
- income < W
- Y is Z * 3000
- W is 9000 + 2500*Z
- Client has Z dependents
- income is steady
- income is not steady

Problem decomposition into an and-or graph

- Pieces of evidence describing the current state of affairs appear in the bottom layer of the graph.

- In some cases, these are statements that may or may not be true, depending on the features of the case currently under consideration.

Problem decomposition into an and-or graph

- In the next layer up, there are simple operations such as calculations and comparisons.

- The lines indicate which pieces of evidence act is inputs to these operations.

Problem decomposition into an and-or graph

- In the upper layers, there are conclusions that can be drawn, if the pieces of evidence (and the results of the simple operations) feeding into them (from below) are true.

- Above them, there are further conclusions that can be drawn if the conclusions feeding into them are true.

Problem decomposition into an and-or graph

- At the top, we have the final conclusion of the reasoning process.

- The variables W, Y and Z allow this graph to represent both numerical and logical reasoning. The variable X allows it to deliver more than one possible conclusion.

Problem decomposition into an and-or graph

- Students often seem to mix up this sort of chart with a flow-chart. It isn't the same thing at all.

  - A flow-chart shows the sequence of operations that a program must go through (probably with some branching) to execute its task.

  - The and-or graph shows the pieces of evidence that must be present if a certain conclusion is to be reached.

Problem decomposition into an and-or graph

- Why draw and-or chart?

- Because it makes the point that a decision-making process like this can be broken down into a sequence of simple decisions, in a very systematic way.

- Because it's a first step towards turning a human being's reasoning into a collection of production rules.
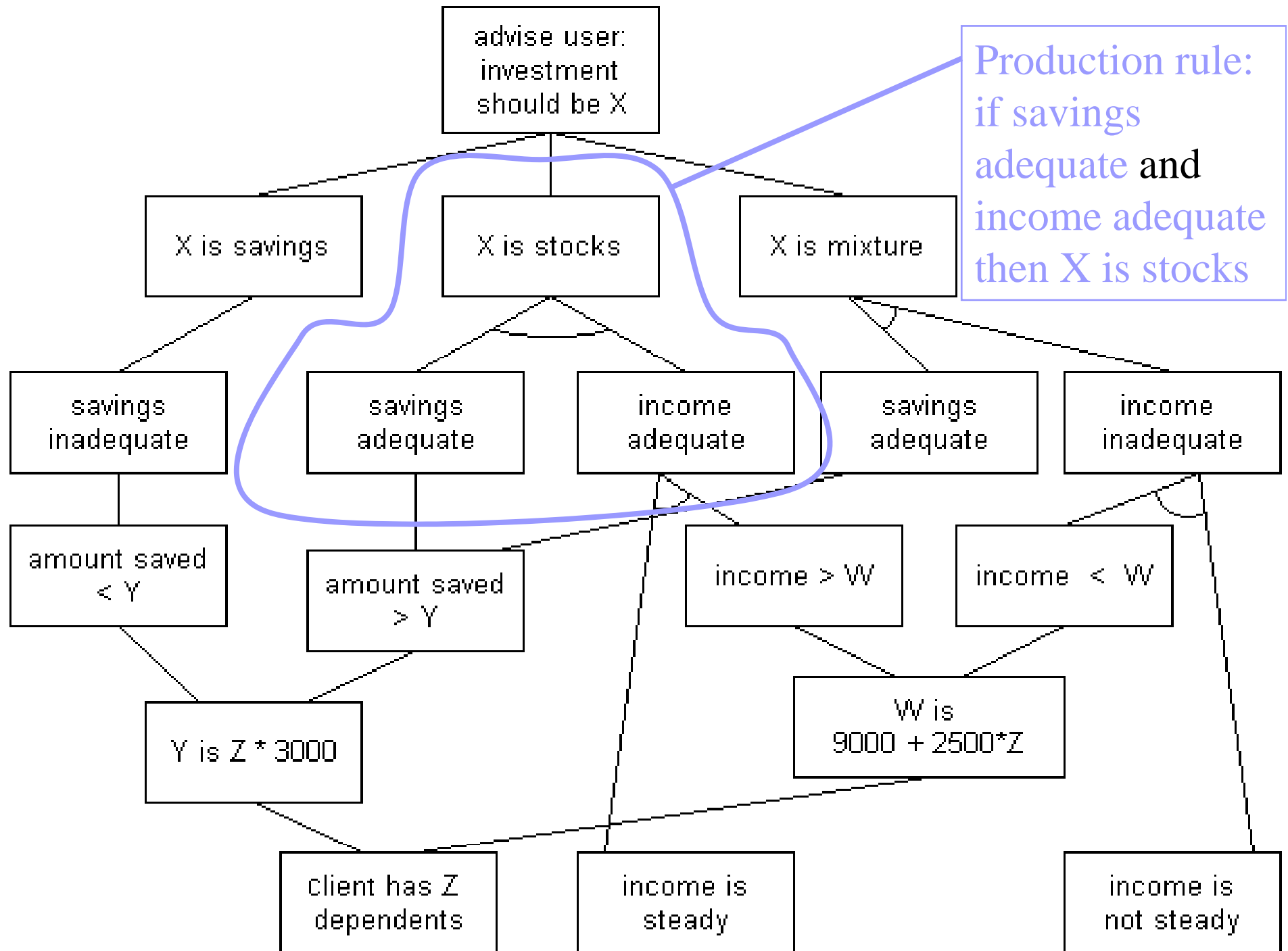
Problem decomposition into an and-or graph

- The and-or chart can be considered as a backward-chaining production system, reading from the top to the bottom. Or as a forward-chaining production system, reading from the bottom to the top.
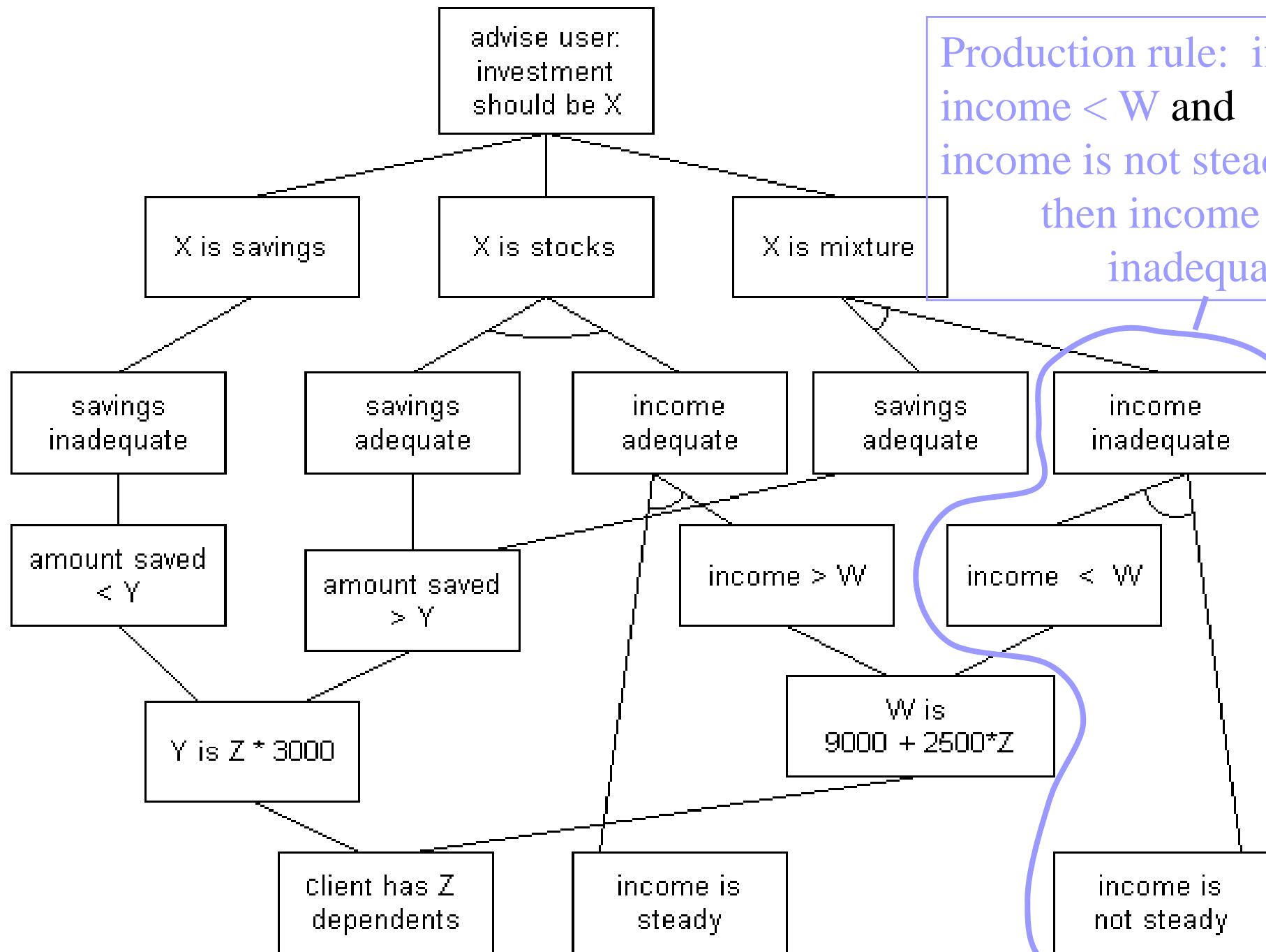
Problem decomposition into an and-or graph

- Every node is the conclusion of a production rule, except for the leaf nodes at the bottom, which are requests for information from the user.

  - When several links enter a node from below, they represent the conditions for that production rule. They may be joined by "and" connectives or by "or" connectives.

  - Alternatively, if several links (or groups of links) enter a node from below, and they are "or" links (or groups of links separated by "or"), this can represent several different production rules which all have the same conclusion.
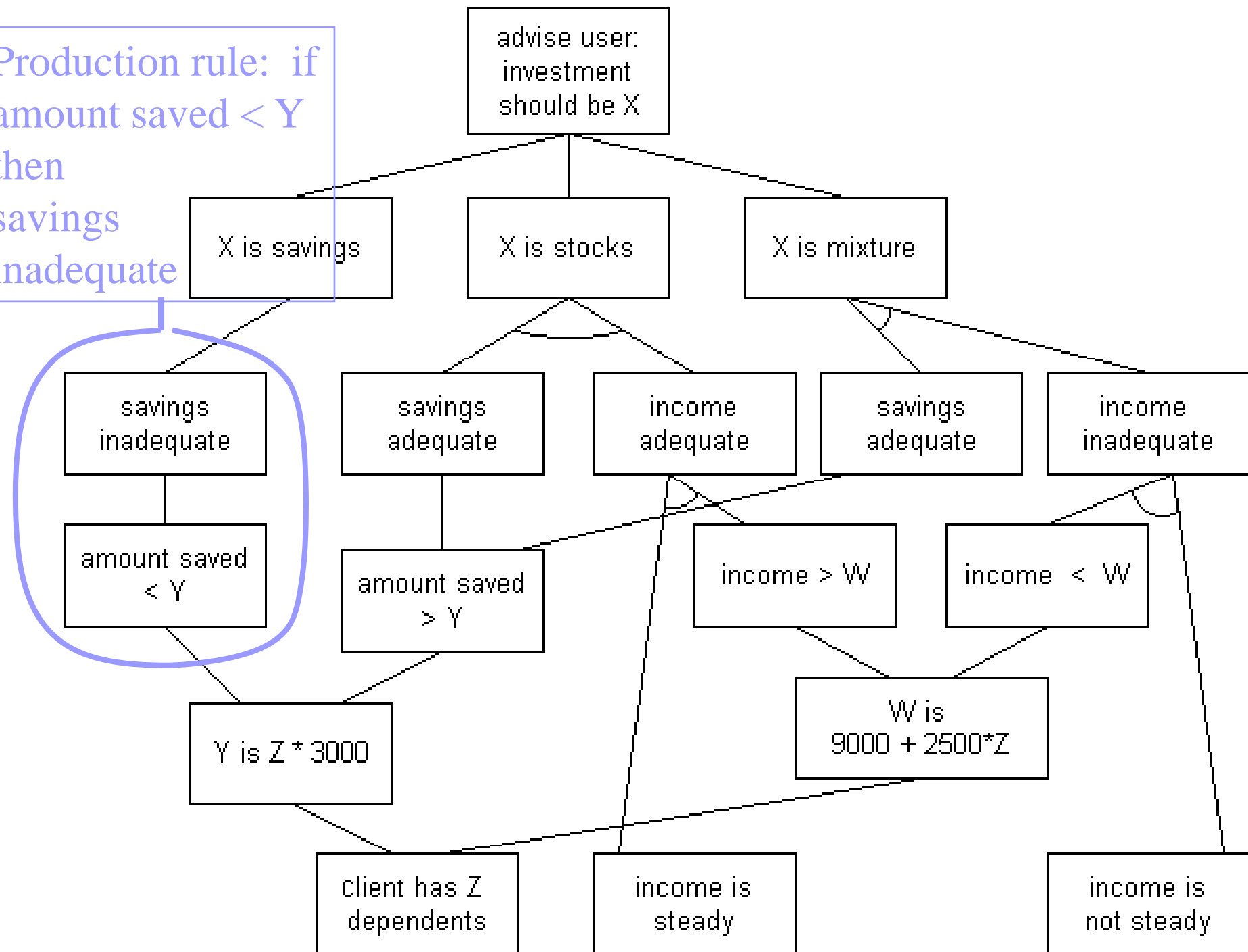
advise user: investment should be X

X is savings

X is stocks

X is mixture

Production rule: if savings adequate and income adequate then X is stocks

savings inadequate

savings adequate

income adequate

savings adequate

income inadequate

amount saved < Y

amount saved > Y

income > W

income < W

Y is Z * 3000

W is 9000 + 2500*Z

Client has Z dependents

income is steady

income is not steady

Production rule:
if X is savings
or X is stocks
or X is mixture
then advise
user investment
should be X

advise user:
investment
should be X

X is savings

X is stocks

X is mixture

savings
inadequate

savings
adequate

income
adequate

savings
adequate

income
inadequate

amount saved
< Y

amount saved
> Y

income > W

income < W

Y is Z * 3000

W is
9000 + 2500*Z

Client has Z
dependents

income is
steady

income is
not steady

Problem decomposition into an and-or graph

- The underlying principle is that state spaces can always be converted into production systems, and vice-versa. Searching a large production system is essentially the same problem as searching a large state-space.