University of Applied Sciences Northwestern Switzerland
School of Business

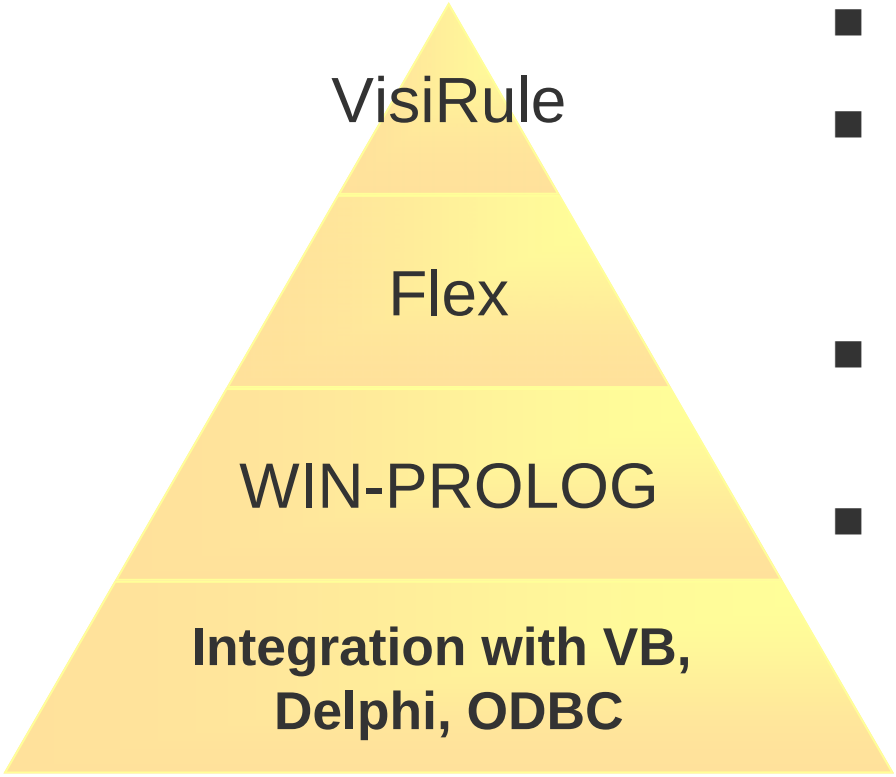
Visual Rule Modelling

- Logic programming is a powerful, declarative programming language
- Prolog is both
  - a programming language with control construct, input/output operations, arithmetics etc.
  - a knowledge representation language
- But writing a knowledge base in Prolog or rule-based systems is not user-friendly
- User-friendly interfaces can be built on top of Prolog or forward-chaining rule systems to represent and modify rule-based knowledge bases

VisiRule – a graphical modelling tool

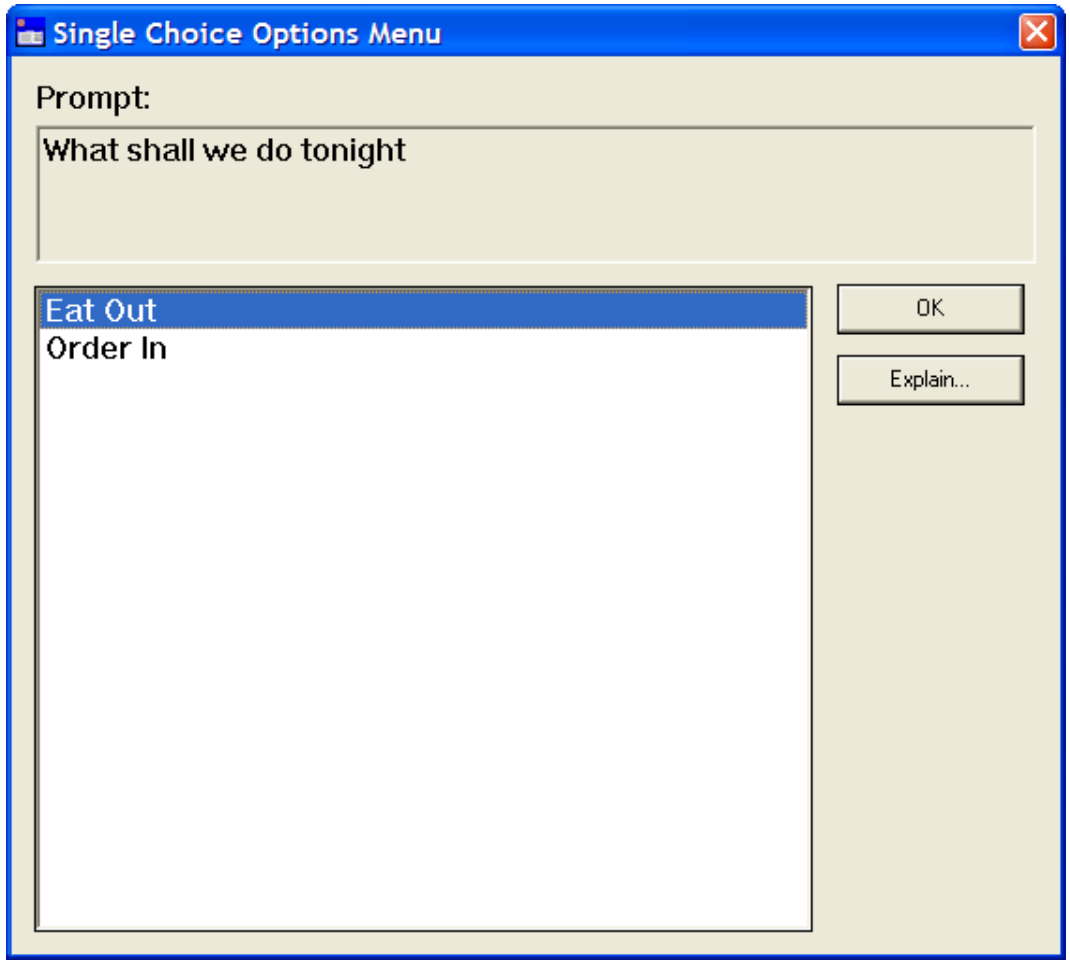
- VisiRule™ is an extension to WIN-PROLOG from Logic Programming Associates Ltd.
- VisiRule allows experts to build decision models using a graphical paradigm
- VisiRule allow to graphically represent forward chaining rules with access to Prolog
- VisiRule models can be interpreted and executed
- VisiRule models can be exported to other programs and integrated into existing web and desktop standards

Layers of VisiRule

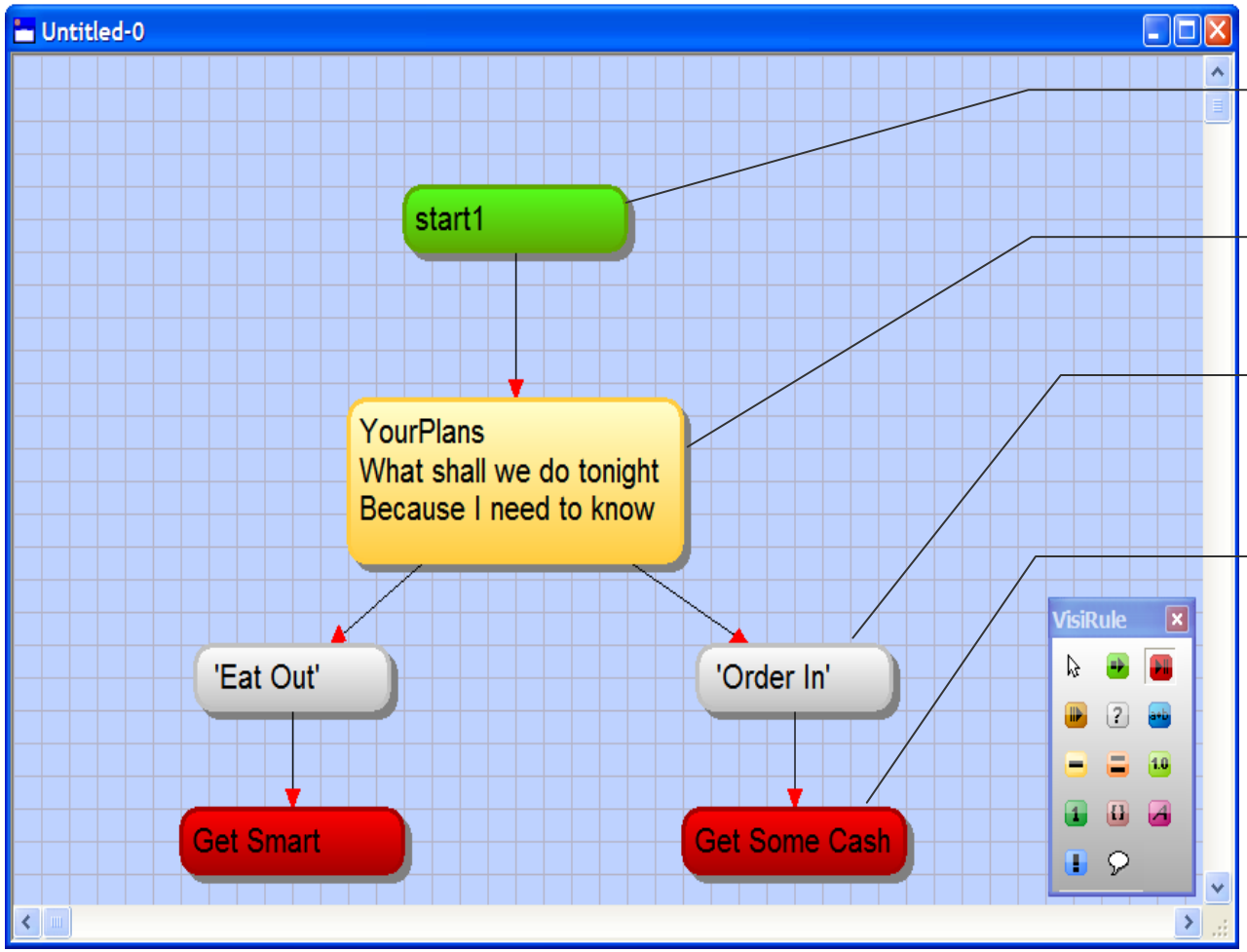


- VisiRule creates charts (layer 1)
- VisiRule generates Flex code (LPA ‘s Expert System Product) (layer 2)
- Flex code in turn generates Prolog (layer 3)
- The underlying Prolog allows to do almost anything, including call C functions using a built-in predicate called winapi/4

Answer the question … and get result



A simple visible chart



Start node

Question

Expressions

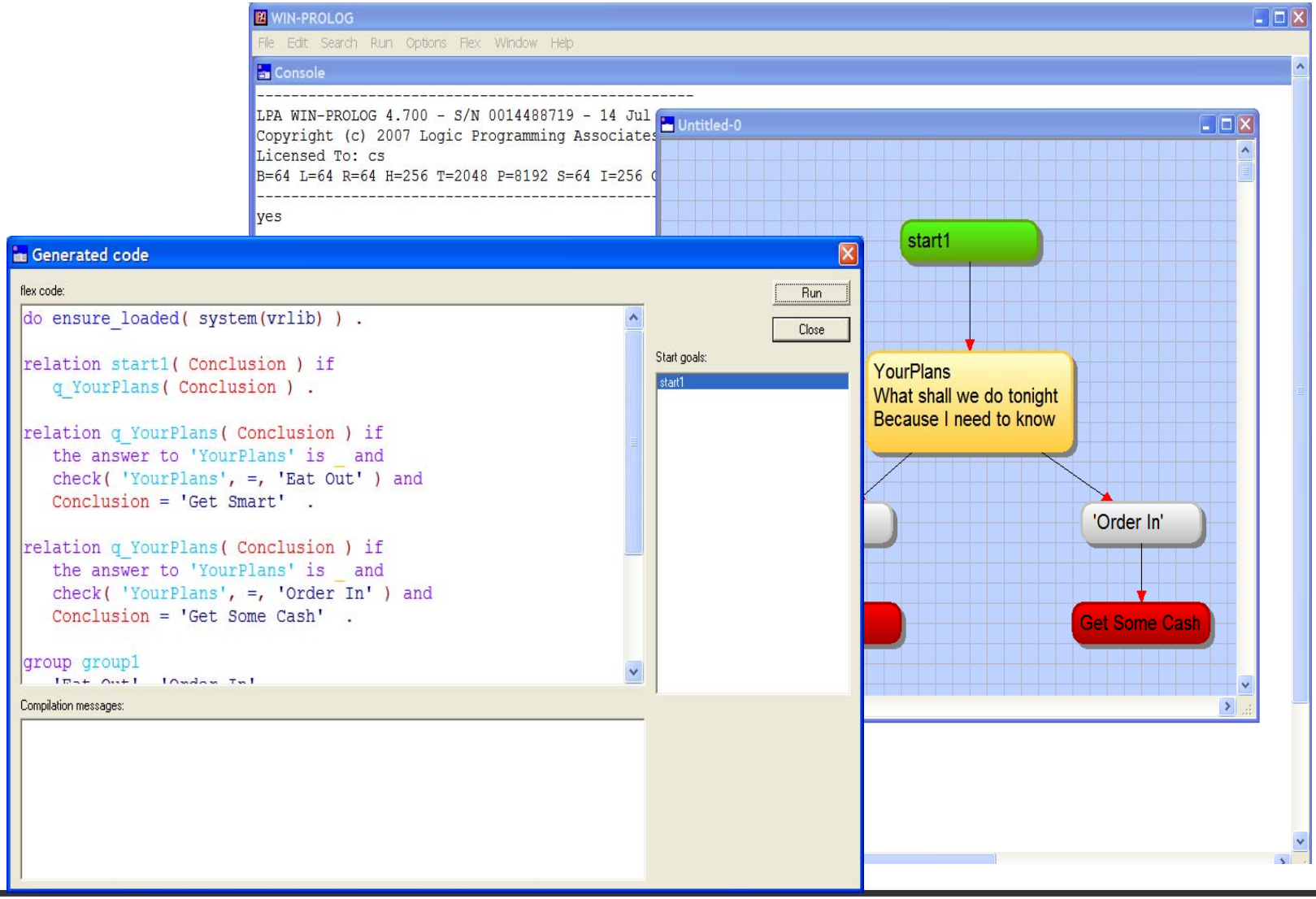
End nodes

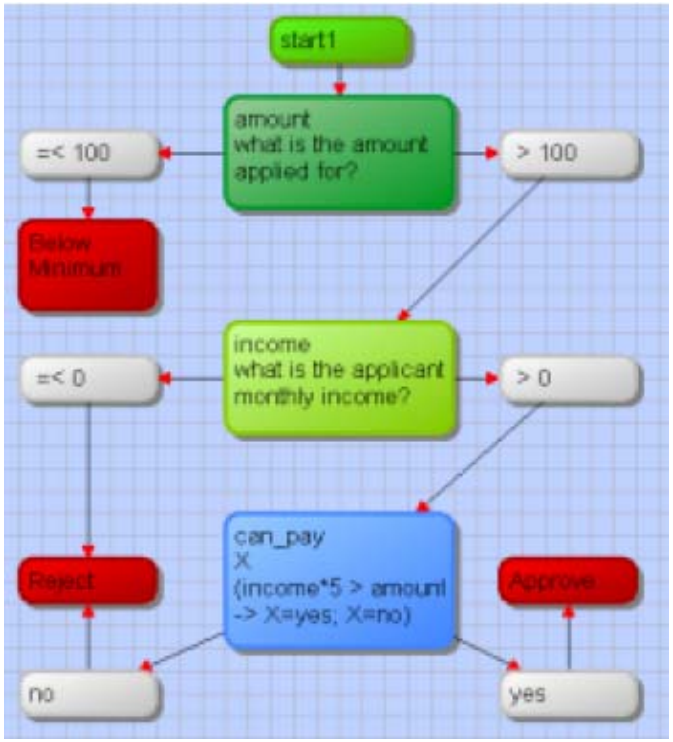
Question Types

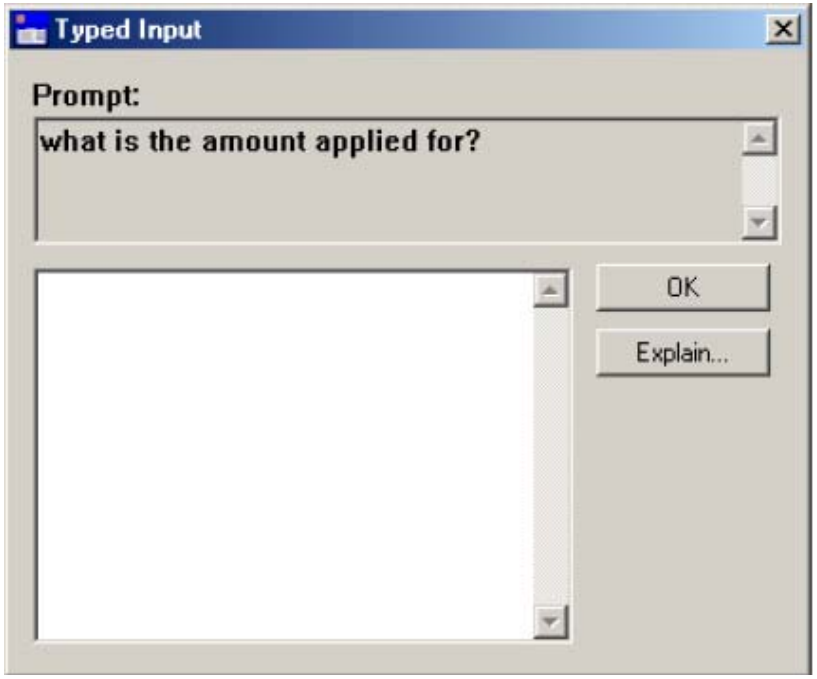
Question Types

Single Choice	This is the default option. The menu produced will only allow the user to select one of the items on the menu.
Multiple Choice	This allows the user to select any or none of the items on the menu.
Number Input	Instead of a menu, this option provides an input box into which the user can enter any number
Integer Input	This is like Number Input, but only allows the user to enter an integer.
Set Input	An input box is also provided by this option. The user can type in a list of items, separated by a space character. For example: red amber green.
Name Input	Another input box is provided into which the user can type a word or phrase.

Generating and running executable code




start1
amount
what is the amount
applied for?
=< 100
> 100
Below
Minimum
income
what is the applicant
monthly income?
=< 0
> 0
can_pay
X
(income*5 > amount
-> X=yes; X=no)
Reject
Approve
no
yes

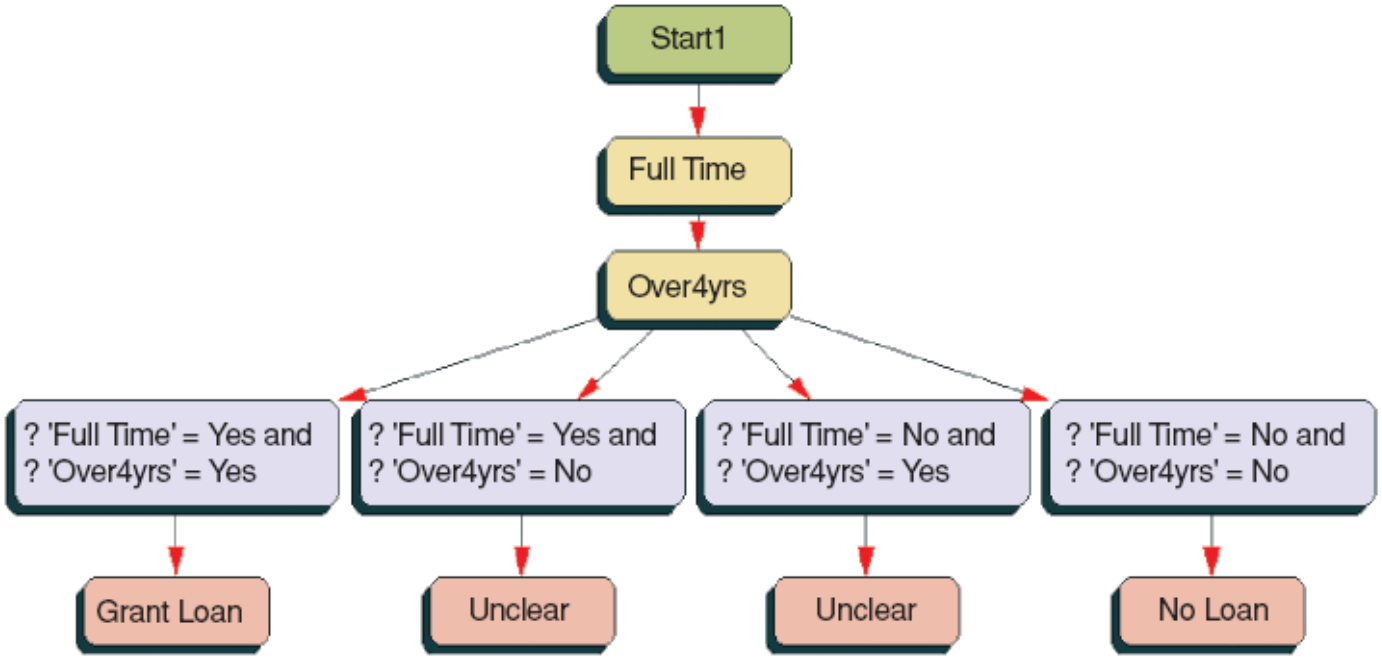

Typed Input
Prompt:
what is the amount applied for?
OK
Explain...

Representing a Decision in VisiRule

VisiRule diagrams are graphical representations of forward chaining rules:

Table 1. Simple set of rules.

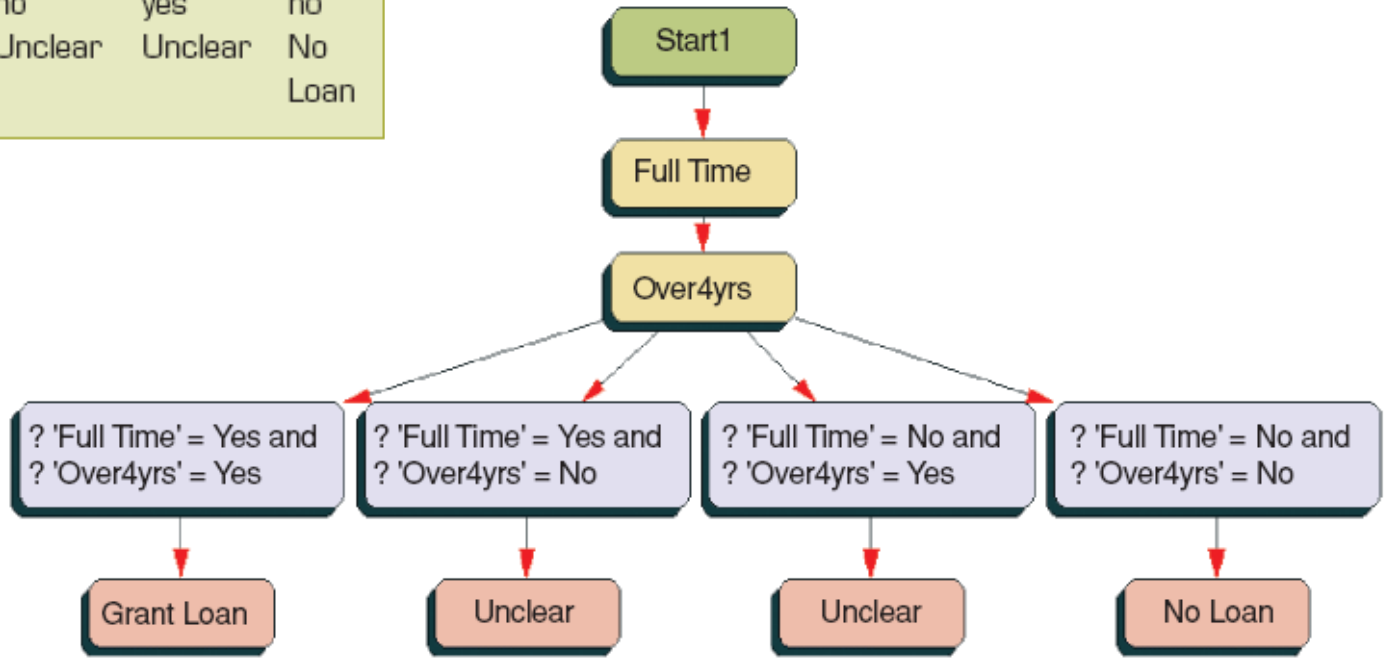
If Full time = yes	and Over4yrs = yes	then answer = Grant Loan
If Full time = yes	and Over4yrs = no	then answer = Unclear
If Full time = no	and Over4yrs = yes	then answer = Unclear
If Full time = no	and Over4yrs = no	then answer = No Loan



The same rule set can also be represented as a decision table:

Table 2. Initial decision table.

Full time	yes	yes	no	no
Over4yrs	yes	no	yes	no
	Grant Loan	Unclear	Unclear	No Loan



Travel Advisory represented as Decision Table and Decision Tree

hot or cold	continent	terrain	destination
cold	n_america	mountains	N/A
cold	n_america	desert	N/A
cold	n_america	snow	Rockies
cold	n_america	sea	Boston
cold	n_america	rivers	Mississippi
cold	n_america	mixed	Victoria
cold	asia	mountains	N/A
cold	asia	desert	N/A
cold	asia	snow	Ladahk
cold	asia	sea	Thailand
cold	asia	rivers	Kashmir
cold	asia	mixed	India
mixed	europe	mountains	Alps
mixed	europe	desert	N/A
mixed	europe	snow	N/A
mixed	europe	sea	Baltic
mixed	europe	rivers	Rhine
mixed	europe	mixed	Ireland
hot	s_america	mountains	N/A
hot	s_america	desert	Venezuala
hot	s_america	snow	N/A
hot	s_america	sea	Bahia
hot	s_america	rivers	Amazon
hot	s_america	mixed	Peru
hot	africa	mountains	N/A
hot	africa	desert	Sahara
hot	africa	snow	N/A
hot	africa	sea	Gambia
hot	africa	rivers	Nile
hot	africa	mixed	Nigeria

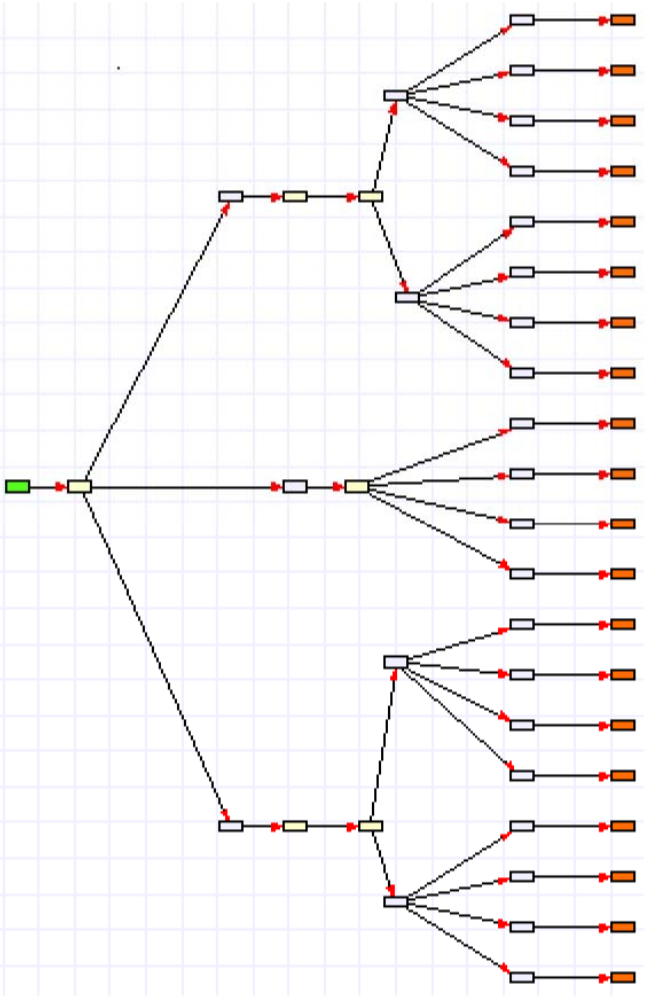
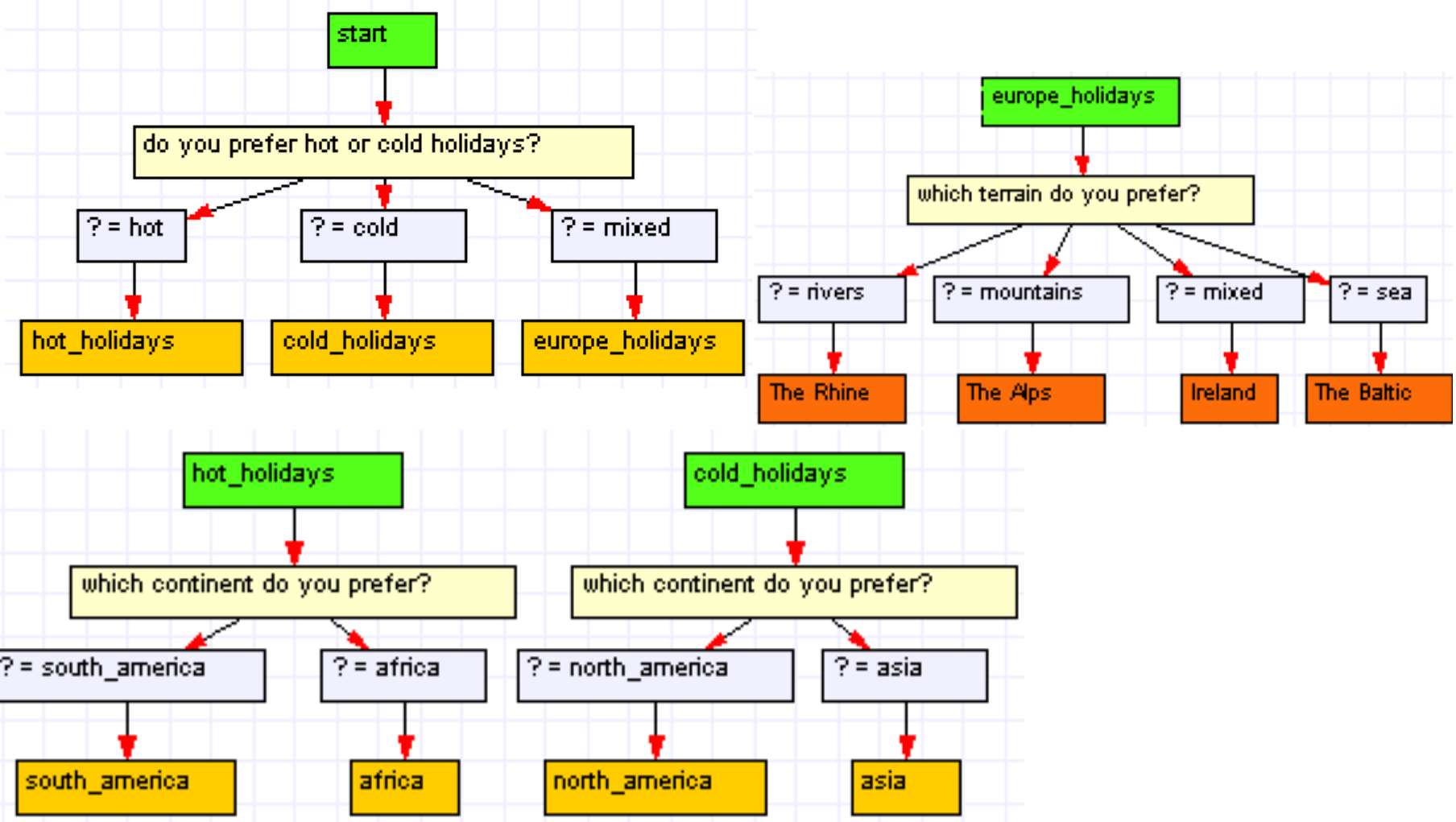
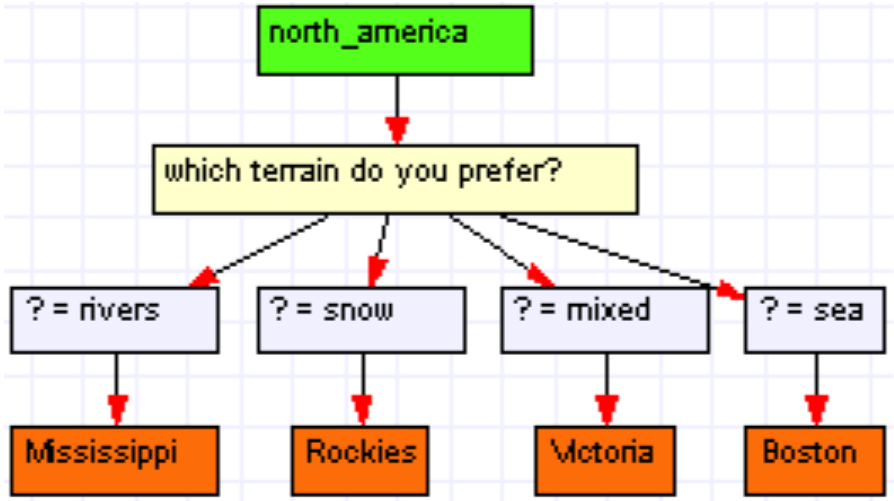


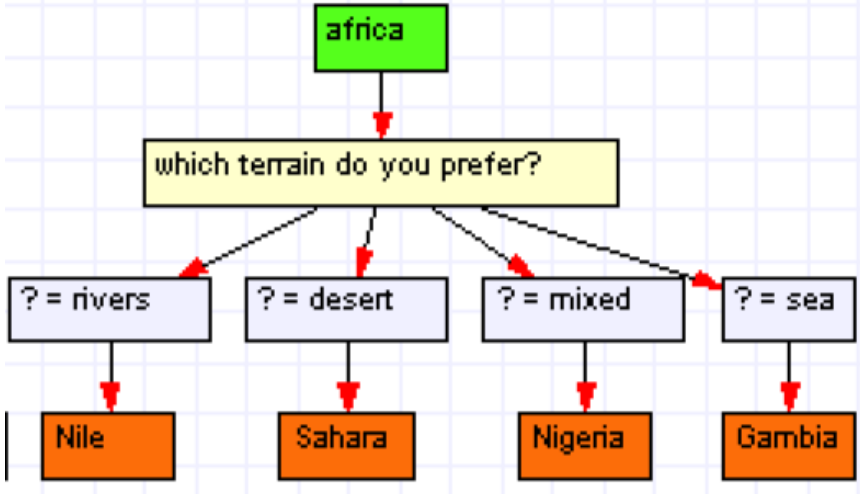
Figure 15 - Two representations

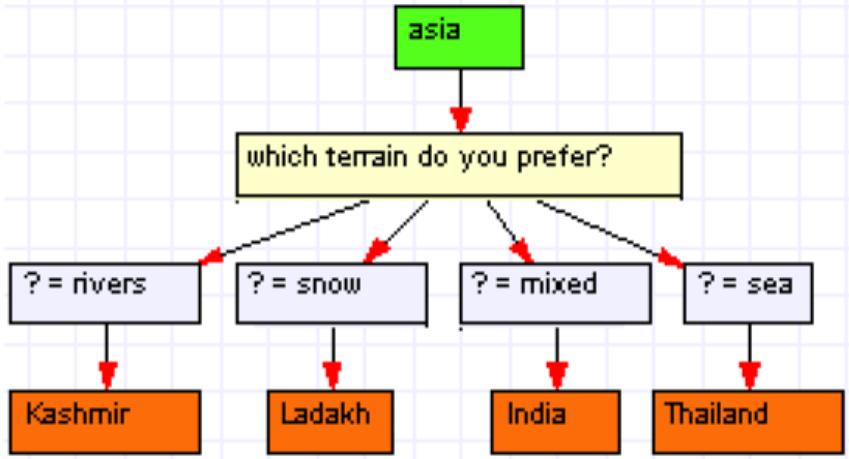
Dividing a Decision Tree into Subtrees using Continuation Boxes

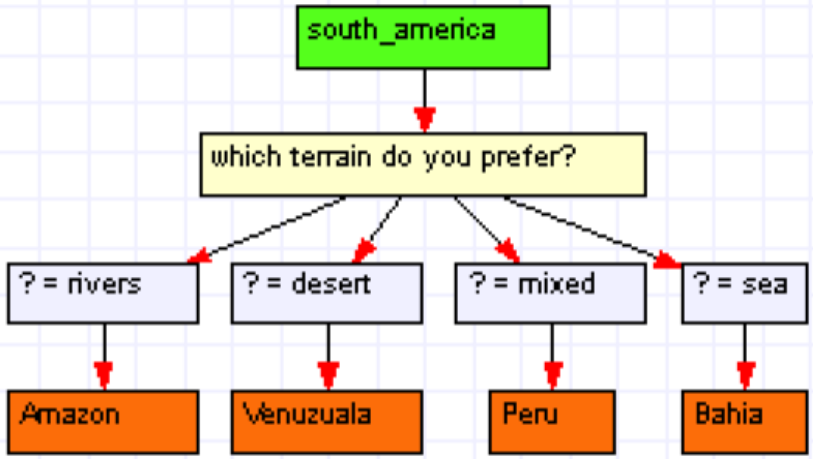


Dividing a Decision Tree into Subtrees using Continuation Boxes (Cont.)



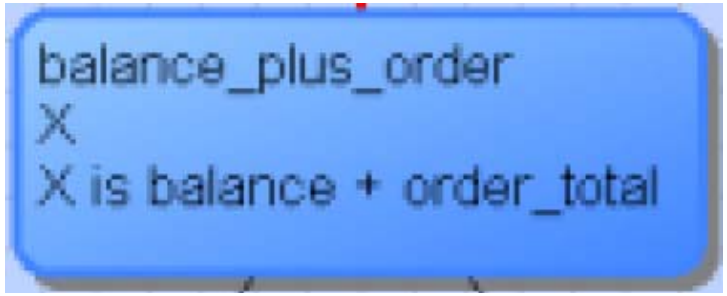




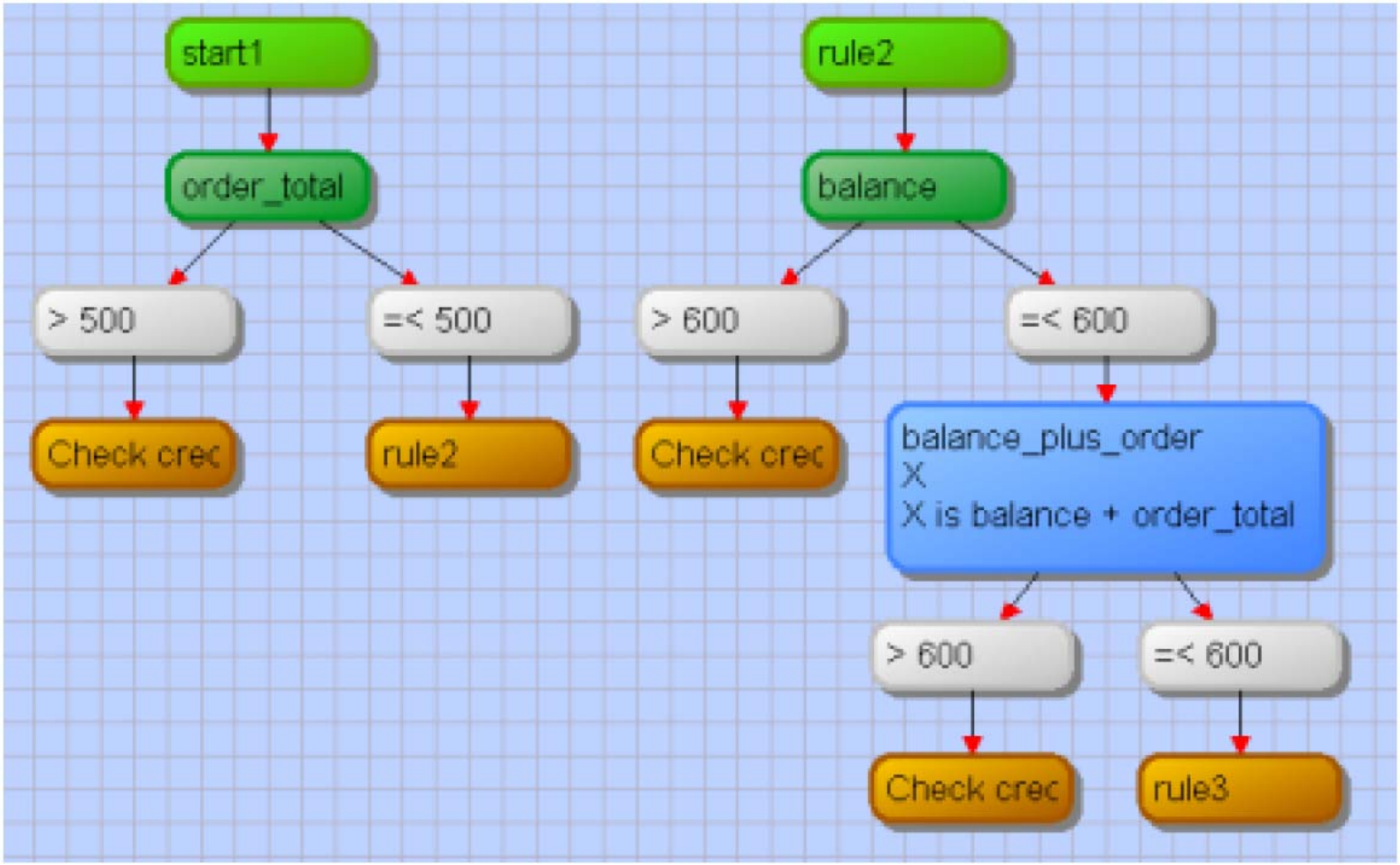


Statement Box

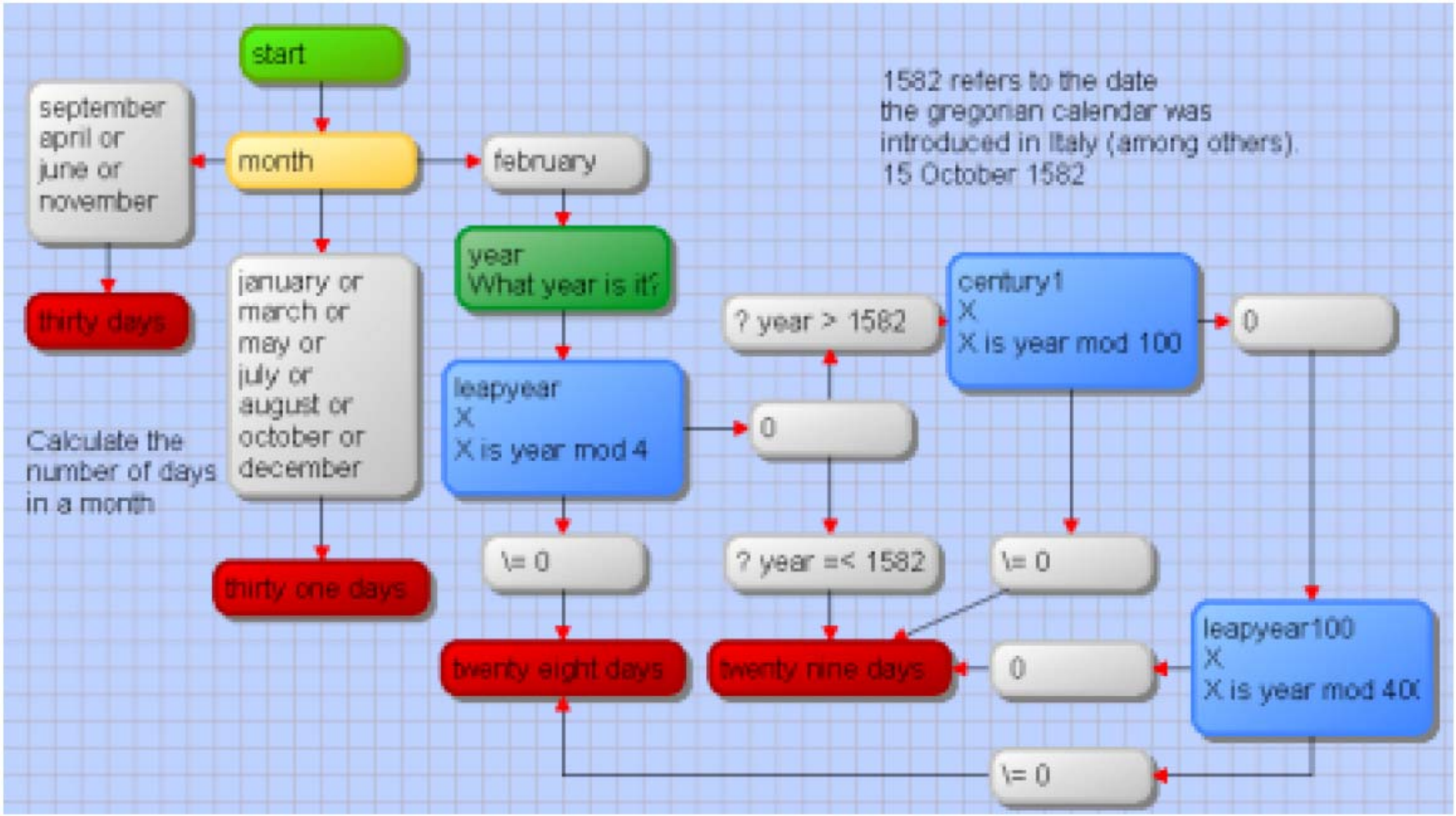
- The function of a statement box is to calculate a value from information that is already known.
- Statement boxes have three elements:
  - an editable ***name*** (balance_plus_order in example below)
  - an editable local ***variable*** (X in example below)
  - editable Prolog code which is used to calculate the value (X is balance+order_total.)



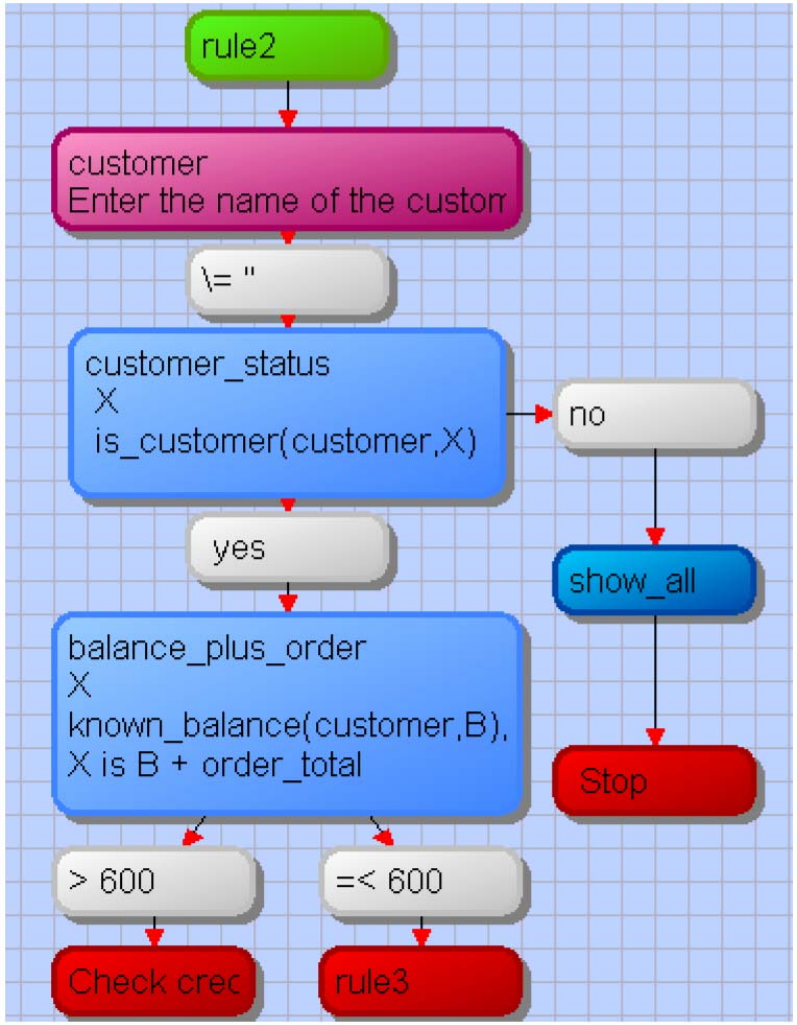
Statement Box with an Arithmetic Expression



Example: Calculating Leap Years



Using a Statement Box to access a Prolog Knowledge Base




```
% customer.pl

known_balance('John Smith',800).
known_balance('Joe Doe',100).
known_balance('Mary Jones',10).

is_customer(X,'yes'):-
    known_balance(X, Bal).
is_customer(X,'no').

show_all:-
    flash('Here is a list of customers'),
    known_balance(X, Bal),
    write(X),nl,
    fail.
show_all.
```