Chapter 5: Simple Modelling

Where we examine various modelling abilities of CLP languages

1

Simple Modelling

- Modelling
- Modelling Choice
- Iteration
- Optimization

2

Modelling

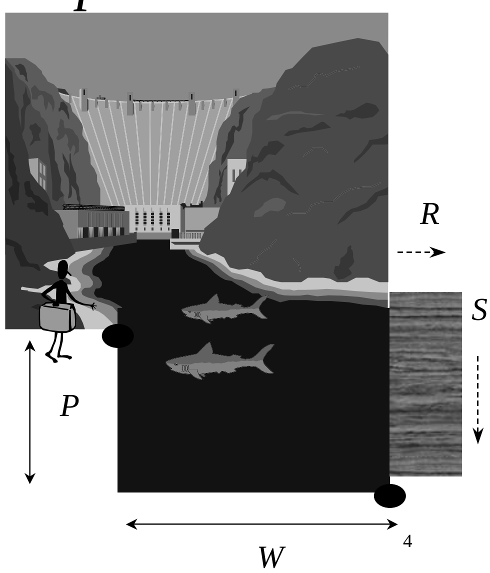
- Choose the variables that will be used to represent the parameters of the problem (this may be straightforward or difficult)
- Model the idealized relationships between these variables using the primitive constraints available in the domain

3

Modelling Example

A traveller wishes to cross a shark infested river as quickly as possible. Reasoning the fastest route is to row straight across and drift downstream, where should she set off

width of river: *W*
speed of river: *S*
set of position: *P*
rowing speed: *R*



4

Modelling Example

Reason: in the time the rower rows the width of the river, she floats downstream distance given by river speed by time. Hence model

```
river(W, S, R, P) :- T = W/R, P = S*T.
```

Suppose she rows at 1.5m/s, river speed is 1m/s and width is 24m.

```
river(24, 1, 1.5, P).
```

Has unique answer $P = 16$

5

Modelling Example Cont.

If her rowing speed is between 1 and 1.3 m/s and she cannot set out more than 20 m upstream can she make it?

```
1 <= R, R <= 1.3, P <= 20,
     river(24,1,R,P).
```

Flexibility of constraint based modelling!

6

Modelling Choice

- Multiple rules allow modelling relationships that involve choice
- E.g. tables of data using multiple facts.

```
father(jim,edward).     mother(maggy,fi).
father(jim,maggy).      mother(fi,lillian).
father(edward,peter).
father(edward,helen).
father(edward,kitty).
father(bill,fi).
```

7

Choice Examples

The goal `father(edward,X)` finds children of Edward. Answers:

$$X = peter,$$
$$X = helen,$$
$$X = kitty$$

The goal `mother(X,fi)` finds the mother of Fi. Answers:

$$X = maggy$$

8

Choice Examples

We can define other predicates in terms of these

```
parent(X,Y) :- father(X,Y).
parent(X,Y) :- mother(X,Y).
sibling(X,Y) :- parent(Z,X), parent(Z,Y),
                X != Y.
cousin(X,Y) :- parent(Z,X), sibling(Z,T),
               parent(T,Y).
```

The goal `cousin(peter, X)` has a single answer *X = fi*

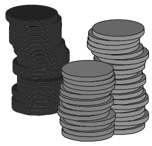
9

More Complicated Choice

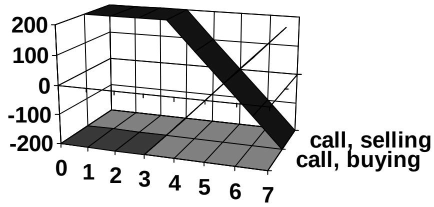
- A **call option** gives the holder the right to buy 100 shares at a fixed price *E*.
- A **put option** gives the holder the right to sell 100 shares at a fixed price *E*
- **pay off** of an option is determined by cost *C* and current share price *S*
- e.g. call cost $200 exercise $300
  - stock price $2, don’t exercise payoff = -$200
  - stock price $7, exercise payoff = $200

10

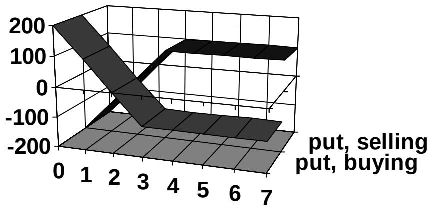
Options Trading

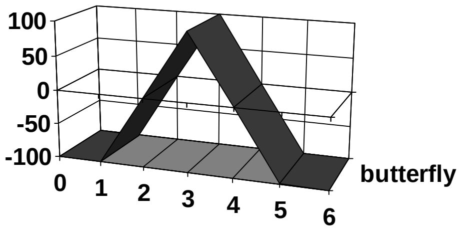


call C=200, E = 300



put C=100, E = 300





Butterfly strike: buy call at 500 and 100 sell 2 puts at 300

11

Modelling Functions

$$call\_payoff(S,C,E) = \begin{cases} -C & \text{if } 0 \le S \le E/100 \\ 100S - E - C & \text{if } S \ge E/100 \end{cases}$$

Model a function with *n* arguments as a predicate with *n+1* arguments. Tests are constraints, and result is an equation.

```
buy_call_payoff(S,C,E,P) :-
     0 <= S, S <= E/100, P = -C.
buy_call_payoff(S,C,E,P) :-
     S >= E/100, P = 100*S - E - C.
```

12

Modelling Options

Add an extra argument *B=1* (buy), *B = -1* (sell)

```
call_option(B,S,C,E,P) :-
     0 <= S, S <= E/100, P = -C * B.
call_option(B,S,C,E,P) :-
     S >= E/100, P = (100*S - E - C)*B.
```

The goal (the original call option question)

```
call_option(1, 7, 200, 300, P)
```

has answer *P = 200*

13

Using the Model

```
butterfly(S, P1 + 2*P2 + P3) :-
     Buy = 1, Sell = -1,
     call_option(Buy, S, 100, 500, P1),
     put_option(Sell, S, 200, 300, P2),
     call_option(Buy, S, 400, 100, P3).
```

Defines the relationship in previous graph

```
P >= 0, butterfly(S,P).
```

has two answers

$$P = 100S - 200 \wedge 2 \leq S \wedge S \leq 3$$

$$P = -100S + 400 \wedge 3 \leq S \wedge S \leq 4$$

14

Modelling Iteration

- Natural model may be iterating over some parameter
- CLP languages have no direct iteration constructs (**for**, **while**) instead recursion

15

Iteration Example

Mortgage: principal *P*, interest rate *I*, repayment *R* and balance *B* over *T* periods

Simple Interest: $B = P + P \times I - R$

Relationship over 3 periods:

$$P_1 = P + P \times I - R \wedge$$
$$P_2 = P_1 + P_1 \times I - R \wedge$$
$$P_3 = P_2 + P_2 \times I - R \wedge$$
$$B = P_3$$

Number of constraints depend on the variable *T*

16

Reason Recursively

Zero time periods then *B = P*

else new princ. *P + P*I - R* and new time *T-1*

```
mortgage(P,T,I,R,B) :- T = 0, B = P. (M1)
mortgage(P,T,I,R,B) :- T >= 1,
     NP = P + P * I - R, NT = T - 1, (M2)
     mortgage(NP,NT,I,R,B).
```

17

Example Derivation

$$\langle mortgage(P,3,I,R,B) | true \rangle$$
$$\Downarrow M2$$
$$\langle mortgage(P_1,2,I,R,B) | P_1 = P + P \times I - R \rangle$$
$$\Downarrow M2$$
$$\langle mortgage(P_2,1,I,R,B) | P_1 = P + P \times I - R \wedge P_2 = P_1 + P_1 \times I - R \rangle$$
$$\Downarrow M2$$
$$\langle mortgage(P_3,0,I,R,B) | P_1 = P + P \times I - R \wedge P_2 = P_1 + P_1 \times I - R \wedge P_3 = P_2 + P_2 \times I - R \rangle$$
$$\Downarrow M1$$
$$\langle [] | P_1 = P + P \times I - R \wedge P_2 = P_1 + P_1 \times I - R \wedge P_3 = P_2 + P_2 \times I - R \wedge B = P_3 \rangle$$

18

Translating Iteration

- Novice CLP programmers may have difficulty defining recursive relationships
- Give a procedural definition
- translate iteration to recursion
- translate tests and assignments to constraints

19

Translation Example

Pseudo C code for the mortgage problem

```
float mg1(float P, int T, float I, float R)
{
     while (T >= 1) {
          P = P + P * I - R;
          T = T - 1;
     }
     return P;
}
```

Remove the while loop using recursion

20

Translation Example

Pseudo C code for the mortgage problem

```
float mg2(float P, int T, float I, float R)
{
     if (T >= 1) {
          P = P + P * I - R;
          T = T - 1;
          return mg2(P, T, I, R); }
     else
          return P;
}
```

Make each variable only take one value

21

Translation Example

Pseudo C code for the mortgage problem

```
float mg3(float P, int T, float I, float R)
{
     if (T >= 1) {
          NP = P + P * I - R;
          NT = T - 1;
          return mg3(NP, NT, I, R); }
     else
          return P;
}
```

Replace the function with a procedure answer by ref.

22

Translation Example

Pseudo C code for the mortgage problem

```
mg4(float P,int T,float I,float R,float *B)
{
     if (T >= 1) {
          NP = P + P * I - R;
          NT = T - 1;
          mg4(NP, NT, I, R, B);  }
     else
          *B = P;
}
```

Replace tests and assignments by constraints

23

Translation Example

Pseudo C code for the mortgage problem

```
mg(P, T, I, R, B)
     :-
          T >= 1,
          NP = P + P * I - R,
          NT = T - 1,
          mg(NP, NT, I, R, B).
mg(P, T, I, R, B) :- T = 0, (note extra)
          B = P.
```

24

Why Constraints and not C

- Both programs can answer the goal
- `mortgage(500, 3, 10/100, 150, B).`
- But the CLP program can answer
- `mortgage(P, 3, 10/100, 150, 0).`

$$P = 373.028$$

- an even the goal
- `mortgage(P, 3, 10/100, R, B).`

$$P = 0.38553B + 6.14457R$$

25

Optimization

- Many problems require a “best” solution
- **minimization literal**: *minimize(G,E)*
- answers are the answers of goal *G* which minimize expression *E* (in context of state)

26

Optimization Examples

```
p(X,Y) := X = 1.
p(X,Y) :- Y = 1.
```

`X >= 0, Y >= 0, minimize(p(X,Y), X+Y)`

Answers: $X = 1 \wedge Y = 0$ and $X = 0 \wedge Y = 1$

`X >= 0, X >= Y, minimize(true, X-Y)`

Answer: $X >= 0 \wedge X = Y$

`minimize(butterfly(S,P), -P)`

Answer: $S = 3 \wedge P = 100$

27

Optimization Evaluation

- A valuation *v* is a **solution** to a state if it is a solution of some answer to the state
- **minimization derivation step**: *<G1 | C1>* to *<G2 | C2>* where *G1 = L1,L2,...,Lm*
  - *L1* is *minimize(G,E)*
  - exists solution *v* of *<G | C1>* with *v(E) = m* and for all other sols *w*, *m <= w(E)*
    - *G2* is *G,L2,...,Lm* and *C2* is *C1 /\ E = m*
  - else *G2* is [] and *C2* is *false*

28

Optimization Example

```
X >= 0, minimize(X >= Y, X-Y)
```

$$\langle X \geq 0, \text{minimize}(X \geq Y, X - Y) | \mathit{true} \rangle$$
$$\Downarrow$$
$$\langle \text{minimize}(X \geq Y, X - Y) | X \geq 0 \rangle$$
$$\Downarrow$$
$$\langle X \geq Y | X \geq 0 \wedge X - Y = 0 \rangle$$
$$\Downarrow$$
$$\langle [] | X \geq 0 \wedge X - Y = 0 \wedge X \geq Y \rangle$$

$$\langle X \geq Y | X \geq 0 \rangle$$
$$\Downarrow$$
$$\langle [] | X \geq 0 \wedge X \geq Y \rangle$$

Minimum value of X-Y is 0 e.g. $\{X \mapsto 3, Y \mapsto 3\}$

Simplified $X \geq 0 \wedge X = Y$

29

Optimization

Optimization doesnt only have to be at the goal

```
straddle(S,C1+C2,E,P1+P2) :-
     Buy = 1,
     call_option(Buy, S, C1, E, P1),
     put_option(Buy, S, Cs, E, P2).

best_straddle(C,E,P) :-
     minimize(straddle(S,C,E,P),-P).
```

30

Simple Modelling Summary

- Converting problem constraints to constraints of the domain
- Choice is modelled with multiple rules
- Functions are modelled as predicates with an extra argument
- Iteration is modelled using recursion
- Optimization requires a new kind of literal

31