ADM + PARALLELISM

Prospects for parallelisation

" ... two deep flaws of existing languages: *(Baldwin)*
1) reliance on side-effects,
2) use of iteration or recursion to express data parallelism ..."

Definitive scripts mean rationalisation of side-effects

Model nature of the dependency between var values explicitly - NB more expressive than mere constraints

Preserve data relationships in state-based paradigm

Unlike a procedural paradigm: defns are
"assertions about current state"
- cf logic programming / circumscribed state

Current state not the cumulative side-effect of actions

The ordering of a system of definitions is irrelevant

No iterative or recursive constructs incorporated

Why we can't program multiprocessors the way we're trying to do it now

" ... software technology for [general-purpose] parallel programming is in sad shape .... [this paper places] the blame on the **languages** in which we are trying to write parallel programs, and even more fundamentally on the **models of computation** on which those languages are based .... "

Douglas Baldwin

"... no need to distinguish between events initiated by the object and those initiated by some agent outside the object ... avoidance of causality leads to simplification ..." *C A R Hoare*

Interference in parallel programming: *(Baldwin)*

procedural:

"has this variable currently an appropriate value"

functional: *"is this variable currently defined"*

Definitive principles support undefined values

Data dependency

= indication that 2 ops MUST be done sequentially (one produces or destroys a value another needs)

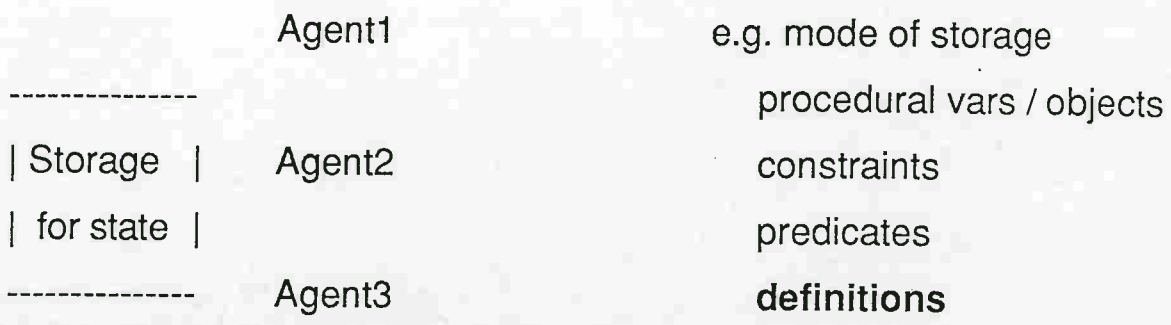
"Ultimate goal for a parallel pl: support clear statement of the data dependencies"

Definitive scripts

encapsulate data dependencies

model indivisible propagation of state-change

Abstract machine model?



What is a definition?

Resembles definition in a functional programming system

MIRANDA script: "**a = f b; b = 17; f x = x*x; c = a + f (f a)**"

Can change definitions only by *editing the script*

[No program can alter the library of programs - **Backus**]

Corresponds to maintenance of data dependencies in a procedural model

Record data dependency

Either update variable values (as in OOP)

or store a recipe for a variable value, evaluate as needed

Compare

"**{a=2} a:=3; b:=f(a); c:=g(b); d:=b*c**"	PROCEDURAL
"**a=2; b=f(a); c=g(b); d=b*c; a=3**"	DEFINITIVE

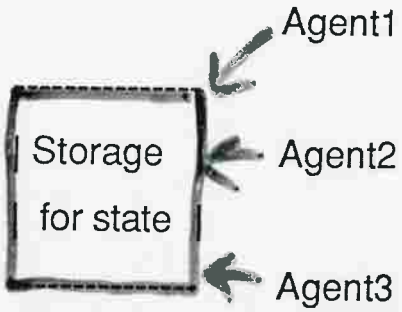
Procedural model:

order significant, VN bottleneck, inefficient if **a** reassigned

Definitive model:

order not significant, needless evaluation avoided, **a** redefined alone

Abstract machine model? . . . . BASIC PRINCIPLES



e.g. mode of storage

TYPICAL
- procedural vars / objects ✗
- constraints ✗
- predicates ✗
- **definitions** ✓

What is a definition?

Resembles definition in a functional programming system

MIRANDA script: "**a = f b; b = 17; f x = x*x; c = a + f (f a)**"

Can change definitions only by *editing the script*

[No program can alter the library of programs - **Backus**]

Corresponds to maintenance of data dependencies in a procedural model

Record data dependency

Either update variable values (as in OOP)

or store a recipe for a variable value, evaluate as needed

Compare

"**{a=2} a:=3; b:=f(a); c:=g(b); d:=b*c**"	PROCEDURAL
"**a=2; b=f(a); c=g(b); d=b*c; a=3**"	DEFINITIVE

Procedural model:

order significant, VN bottleneck, inefficient if **a** reassigned

Definitive model:

order not significant, needless evaluation avoided, **a** redefined alone

N.B. CAPTURES REAL-WORLD SEMANTICS OF E.G. LEGAL TRANSACTIONS

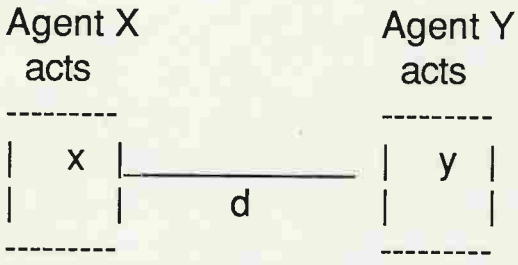
Multi-agent paradigm

Several agents act on a common var system

- can often update in parallel without interference
  definitions <u>don't in general interfere</u> - only if
  - they directly conflict (e.g. "a=b+c" and "a=4")
  - they lead to cyclic dependency (e.g. "a=b+c" and "b=2*a")
  - a sub-exp evaluated in one defn is redefined through another
    (e.g. "a=b+|c+d|" and "c=7")

  [Context of existing var dependencies significant for interference]
- system of defns articulates for each agent (e.g. user of spreadsheet) the current state *and* the expected consequences of actions
  interference expresses the incompatibility of two agent views

An illustrative example



Blocks x and y can move in 1-dimension

Each agent can act
either by **holding** or **releasing** its block, or by **moving** it to R or L

If the connection between the blocks is rigid, then X perceives the context:
"yheld=tt/ff; xheld=tt/ff; y=x+d; x=?"
and Y perceives the context:
" yheld=tt/ff; xheld=tt/ff; x=y-d; y=?"

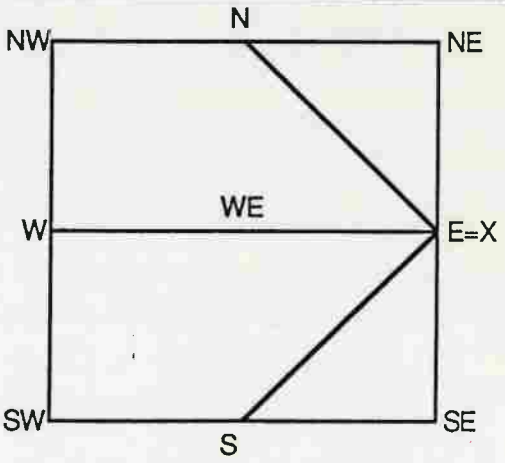
- Concurrent views are inconsistent: "Y can act only if X has released x", etc
- Each agent can only act *conditionally* to redefine *certain* variables cf LSD
- The nature of the connection dramatically affects the perceived context
  e.g. rigid rod vs elastic string that can be cut (cf constraint-based model)

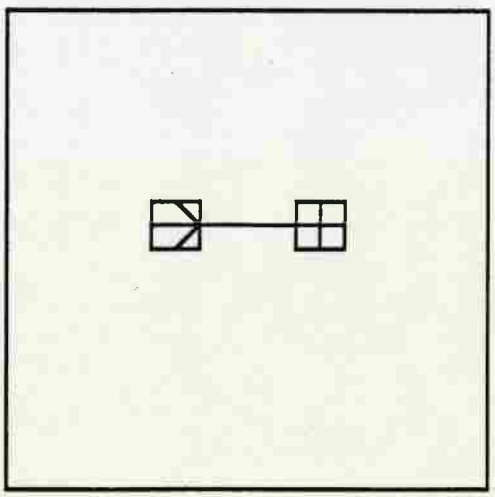
Illustration motivates a machine model in which
system state is represented by a set of defns that can be reconfigured
latent transitions represented by guarded sequences of redefinitions

Figure 1: A block moving simulation using definitive principles

```
entity  handler1()
{
      definition
         d1 = dl1 or dr1,
         dl1 = h1 and pl1, dr1 = h1 and pr1,
          pl1 = 0, pr1 = 0, h1 =0
      action
         not h1 -> h1=1,
          h1 and not d1 -> h1=0: pl1 =1: pr1=1,
         dl1 -> pl1 = 0, dr1 -> pr1 = 0
}

entity  bstate()
{
       definition
          p1, p2, d,
          st = not nostr and (p2-p1)==d,
          t12 = (p2-p1)==1,
          nostr = 0
       action
           not nostr and (p2-p1)>d -> nostr=1
}

entity  bmover()
{
        action
            dl1 and not st -> p1=|p1|-1,
            dl1 and st -> p2=p1+d; p1=|p1|-1,
            dr2 and not st -> p2=|p2|+1,
            dr2 and st -> p1=p2-d; p2=|p2|+1,
            dr1 and not t12 -> p1=|p1|+1,
            dr1 and t12 -> p2=p1+1; p1=|p1|+1,
            dl2 and not t12 -> p2=|p2|-1,
            dl2 and t12 -> p1=p2-1; p2=|p2|-1,
}

bstate(); bmover(), handler1(); handler2()
```

```
openshape b1
within b1 {
    point O
       O = {500+~/p1*100, 500}
     point NE,NW, SW,SE
      NE = O + {50,50}
      NW = O + {50,-50}
     .....
     line  n,s,e,w
      n = [NW,NE]
      s = [SW,SE]
     .....
     point N, E, S, W, X
      N = if ~/h1 then (NE+NW) div 2 else O
      S = if ~/h1 then (SE+SW) div 2 else O
      E = if ~/h1 then (NE+SE) div 2 else O
      W = if ~/h1 then (NW+SW) div 2 else O
      X = if ~/dr1 then E else
                        if ~/dl1 then W else O
      line WE, NX, SX = [W,E], [N,X], [S,X]
}
int   p1,h1,dr1,dl1,p2,h2,dr2,dl2,nostr
line  str
str = [b1/E, if nostr then b1/E else b2/W]
```





Above: The ADM control program

Bottom right: Simulation snapshot

Middle right: Detail of the block1 display

Top right: DoNaLD display specification

Prospects for parallelisation

" ... two deep flaws of existing languages: (Baldwin)
1) reliance on side-effects,
2) use of iteration or recursion to express data parallelism ..."

- Definitive principles involve the rationalisation of side-effects
- Models nature of the dependency between var values explicitly
  - NB more expressive than merely specifying constraints
- Preservation of data relationships within state-based paradigm

- Not primarily a procedural paradigm:
  defns are "assertions about state" - cf logic programming
- Current state not just the cumulative side-effect of actions
- The ordering of a system of definitions is irrelevant
- No iterative or recursive constructs incorporated

AVOID PARADIGM DISCONTIN-UITY

Interference in parallel programming: (Baldwin)
procedural: "has this variable currently an appropriate value"
functional: "is this variable currently defined"

- Definitive principles conveniently support undefined values
- Symbolic manipulation of definitions may have a role
  (c.f. directionality in functional program, where processing may be dependent upon arguments having been evaluated)

Granularity

Indications that can use the ADM at many abstraction levels

? Use hybrid paradigm e.g. to parallelise expression evaluation

Communication

ADM doesn't immediately suggest a multiprocessor allocation

We are exploring a variant of the ADM - "the LSD notation":
actions are associated by agent (vs entity)
communication modelled via shared definitive variables,
loosely synchronised "A redefines x, B later consults"