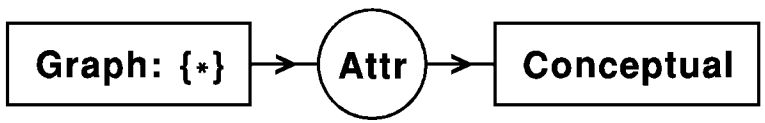
Conceptual Graph Background

David Benn

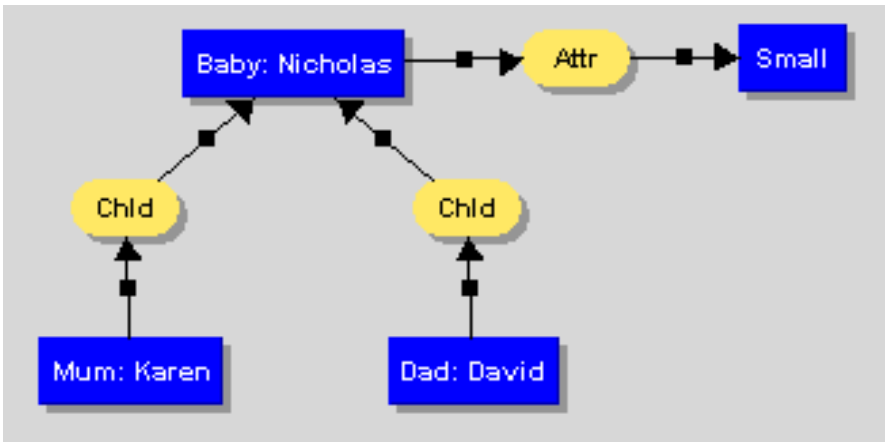
University of South Australia
School of Computer and Information Science

ICCS 2001



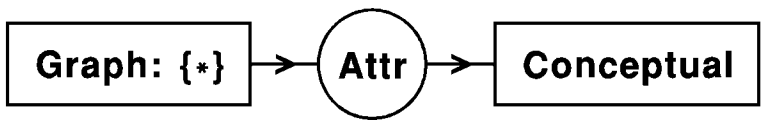
Conceptual Graphs

- **Sowa's CST (1984) permits knowledge representation via bipartite CGs:**
  - **Concepts**
  - **Conceptual Relations, e.g. thematic roles**
  - **Type Lattices**

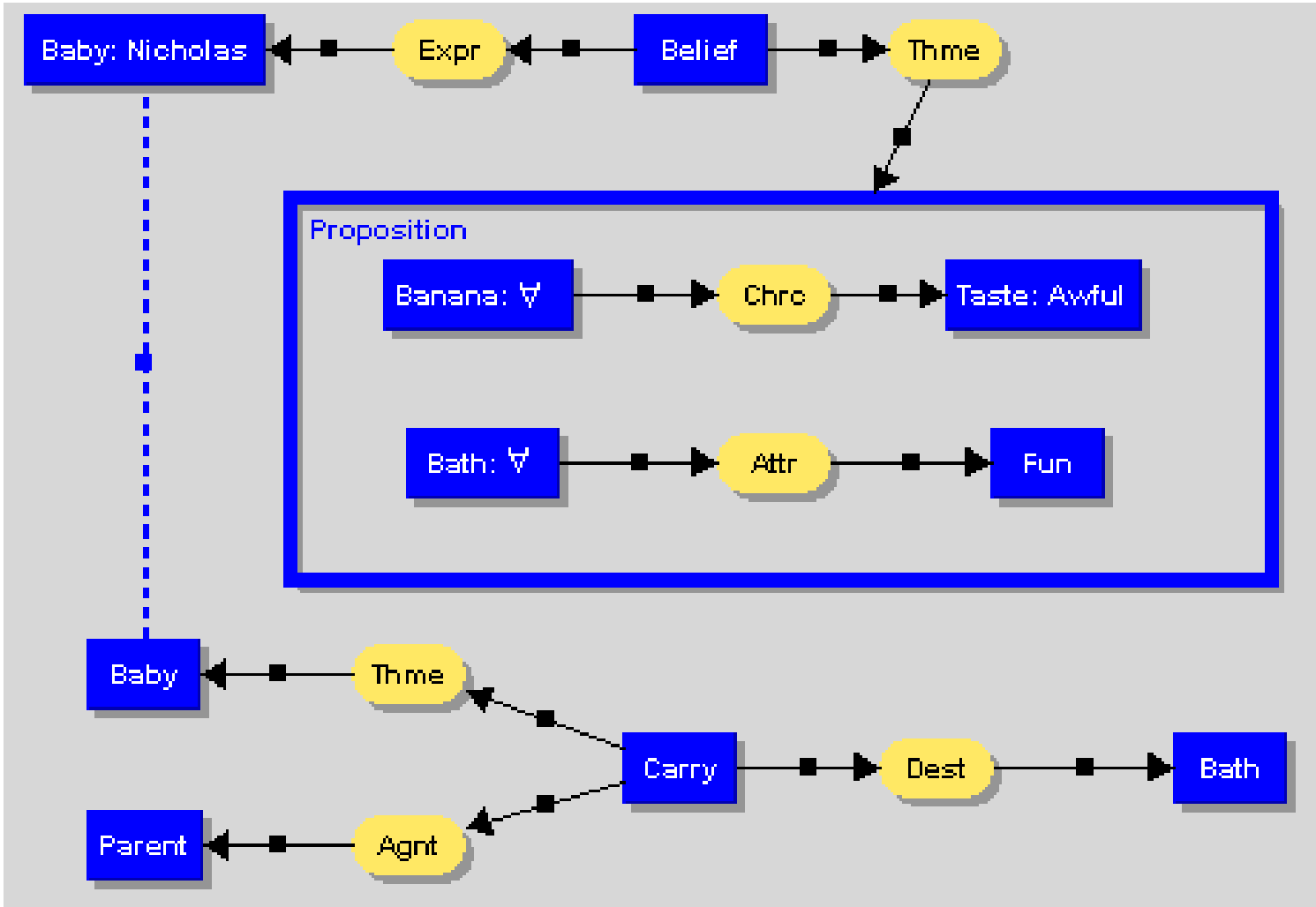



```
(Attr)-
    -> [Small]
    <- [Baby: Nicholas]-
    <-2- (Chld) <-1- [Mum: Karen]
    <-2- (Chld) <-1- [Dad: David]
```

```
(Attr [Baby *a: "Nicholas"] [Small])
(Child ?a [Mum: "Karen"])
(Child ?a [Dad: "David"])
```

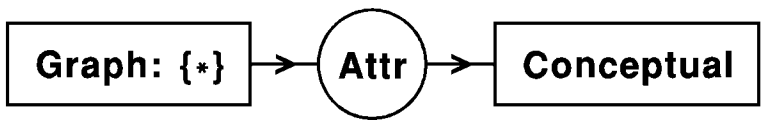
Conceptual Graphs



Graph: {*} → Attr → Conceptual

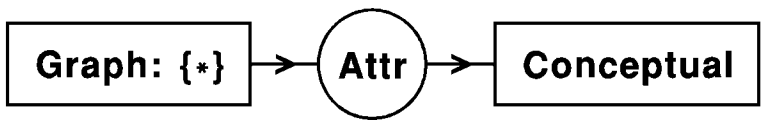
Conceptual Graphs

- **Display and Linear forms, and CGIF.**
- **CST primitives model static knowledge.**
- **Modification of knowledge base (KB) via:**
  - **Copy**
  - ***Restriction***
  - ***Join***
  - **Simplification**
  - ***Projection*** **(π)**



Processes

- **Addressing a lack in CST:**
  - **CGs represent declarative information;**
  - **By themselves they are insufficient for:**
    - **doing computation;**
    - **simulating events and processes.**
  - **Some kind of truth maintenance engine is required: *assertion* and *retraction* of graphs over time.**
  - **Mineau's (1998) CG Processes are one such approach.**

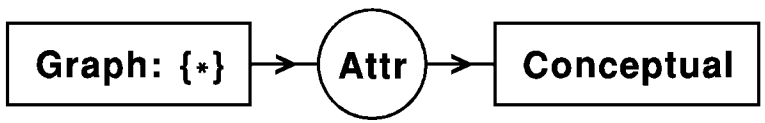


Processes

- **Key notions:**
  - **state transitions;**
  - **knowledge-based *precondition* matching: conjunction of CGs;**
  - ***post-conditions* modify Knowledge Base (KB) by graph assertion and retraction; may use matched information;**
- *process p(in g1, out g2,...) is*

$$\{ r_i = \langle pre_i, post_i \rangle, \forall_i \in [1, n] \}$$

- **forward chaining.**



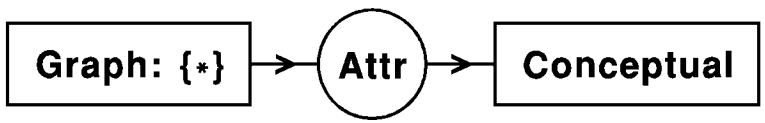
Processes

- **Generalisation of Delugach's (1991) *demons*, and Sowa's *actors* or *dataflow graphs*.**

process > demon > actor

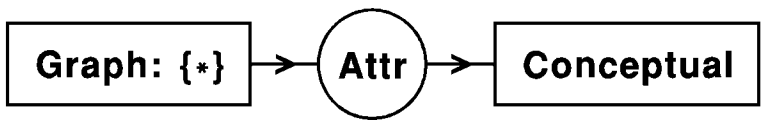
Formalism	Parameters
Processes	Graphs
Demons	Concepts
Actors	Referents

- **Process mechanism also *uses* actors in practice when referent computation required.**

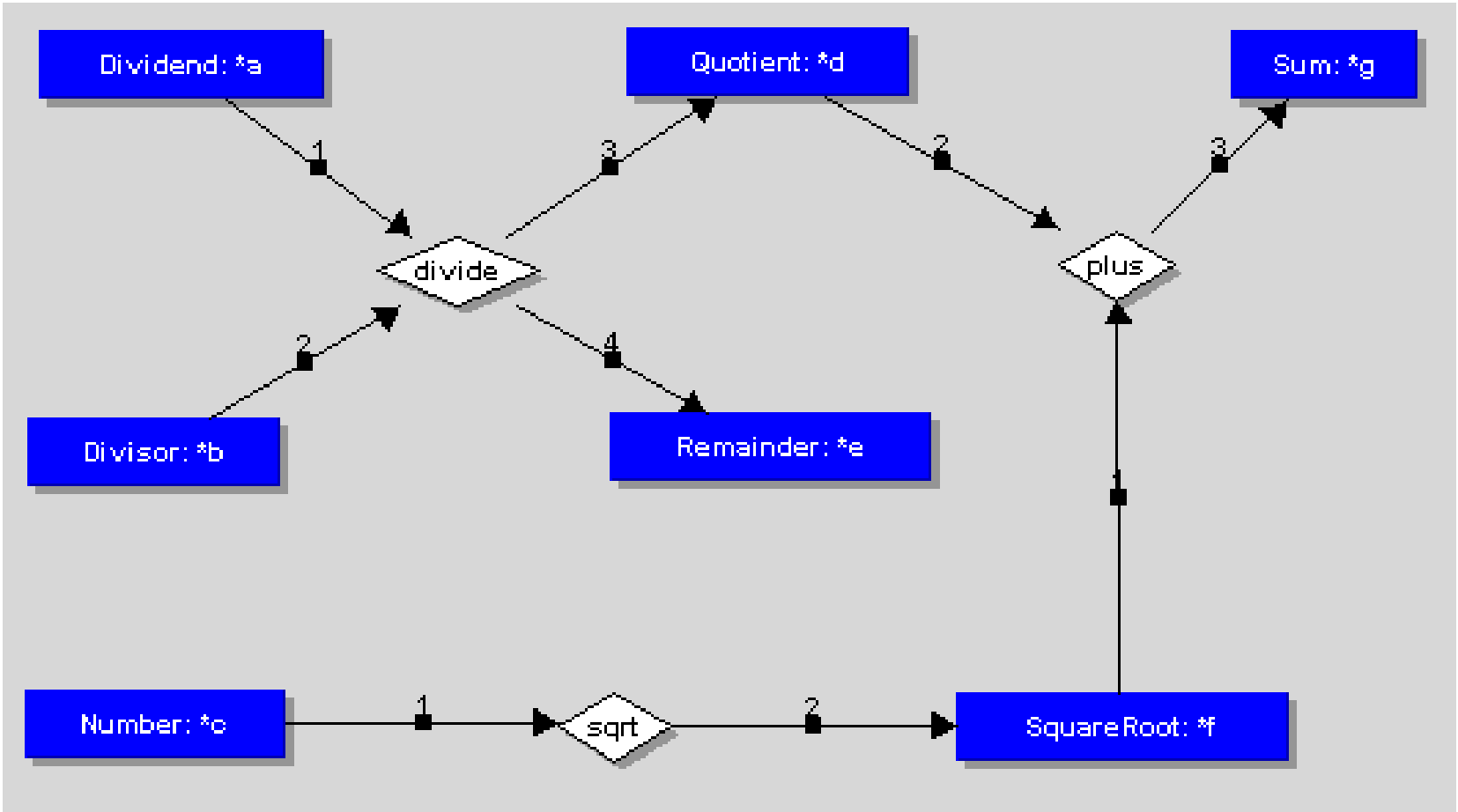


Thesis Content

- **Current work:**
  - **provides a primer on CGs;**
  - **surveys literature re: dynamic CG formalisms & tools, and Petri Nets (coloured, actor mechanism description);**
  - **presents a general-purpose programming language, *pCG*, developed by the author which embodies Mineau's process formalism (`http://www.adelaide.net.au/~dbenn/Masters/`);**
  - **presents and discusses pCG experiments;**
  - **suggests future directions.**



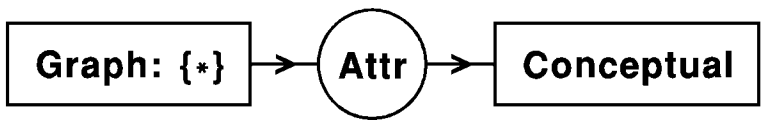
Actors


Dividend: *a
Quotient: *d
Sum: *g
divide
plus
Divisor: *b
Remainder: *e
Number: *c
sqrt
SquareRoot: *f
1
2
3
4
1
2
3
2
1
2

Actors

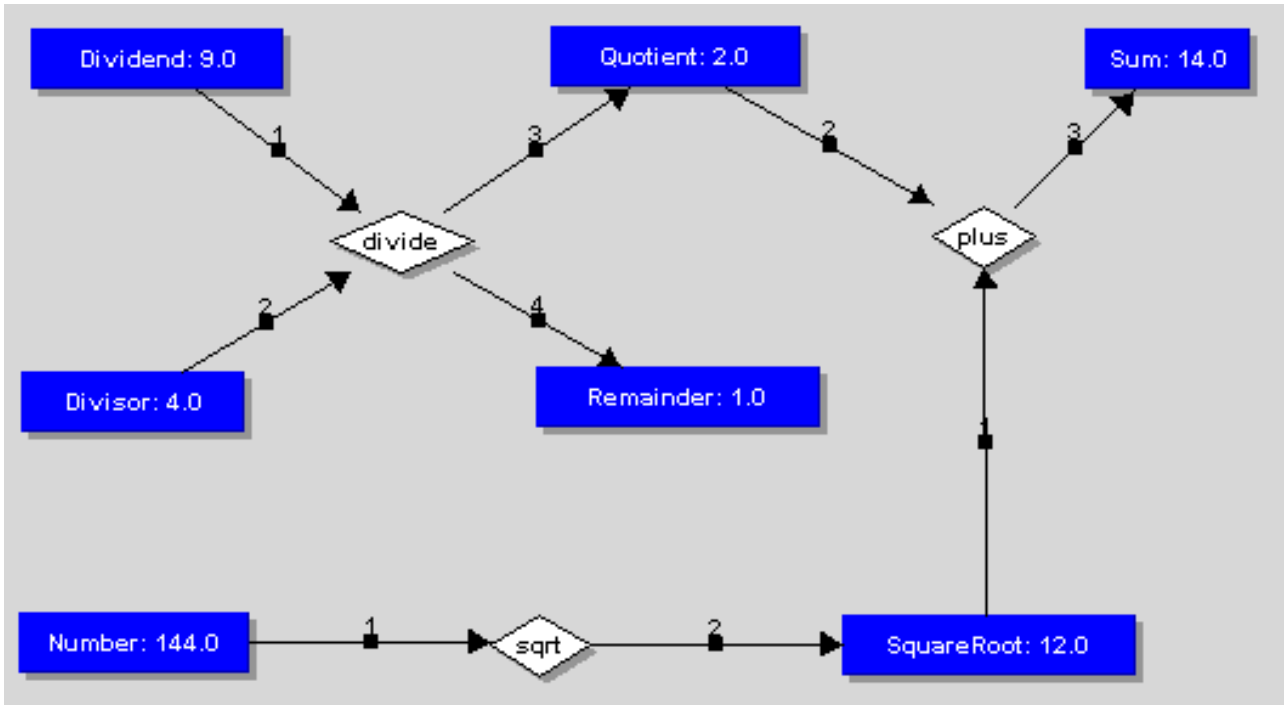
- **CGIF generated by Delugach's CharGer:**

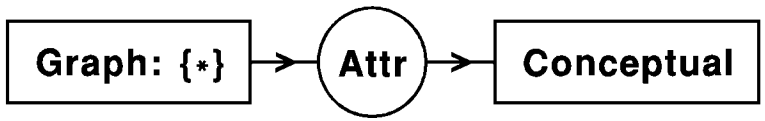
```
[Remainder:*a'*e';CGHSD2.3b&Concept|243|232,190,117,25|&]
[Divisor:*b'*b';CGHSD2.3b&Concept|240|20,192,92,25|&]
[Dividend:*c'*a';CGHSD2.3b&Concept|238|24,51,104,25|&]
[Number:*d'*c';CGHSD2.3b&Concept|234|19,291,95,25|&]
[Sum:*e'*g';CGHSD2.3b&Concept|231|458,51,68,25|&]
[Quotient:*f'*d';CGHSD2.3b&Concept|229|238,50,103,25|&]
[SquareRoot:*g'*f';CGHSD2.3b&Concept|227|358,292,121,25|&]
<divide?c?b|?f?a;CGHSD2.3b&Actor|237|147,126,59,25|&>
<sqrt?d|?g;CGHSD2.3b&Actor|233|213,292,41,25|&>
<plus?g?f|?e;CGHSD2.3b&Actor|226|395,124,42,25|&>/*CGHSD2.3b&G
   raph|225|0,0,40,25|&*/
```

Actors

```
r = file (dir + "Figure1.CGF");
actor Figure1(a,b,c) is r.readGraph();
g = Figure1(9,4,144);
out_path = dir + "Figure1Out.cgf";
w = file (">" + out_path);
w.writeln(g);
```





Results

- **The development of *anonymous recursive actors* in pCG, obviating the need for an actor type definition in the special case of a recursive actor, e.g. in preconditions:**

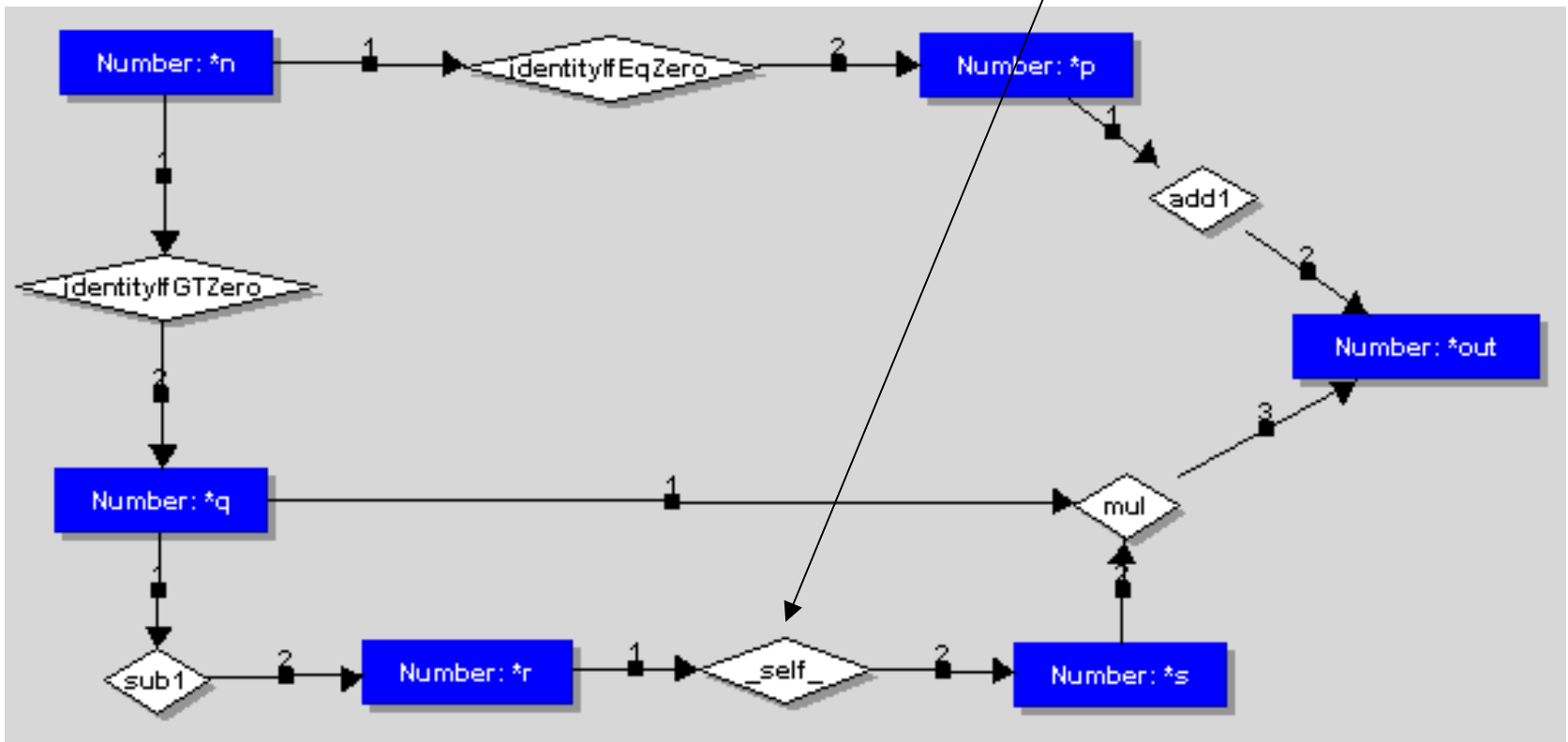
```
function multiply(a,b,result)
  first = a.designator;
  second = b.designator;
  result.designator = first * second;
end
```

```
pre
  ...
  `[Integer:*a'*fValue'][Variable:*b'#f']
   [Integer:*c'*iValue'][Variable:*d'#i']
   [Integer:*e'*product']
     (val?b?a)
     (val?d?c)
     <multiply?a?c|?e>`;
  ...
end
```

Results

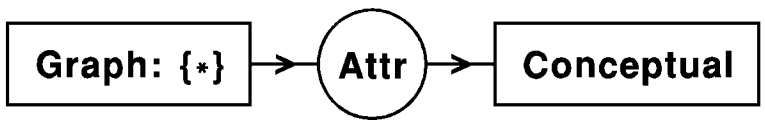
An anonymous recursive actor:

self refers to this defining graph



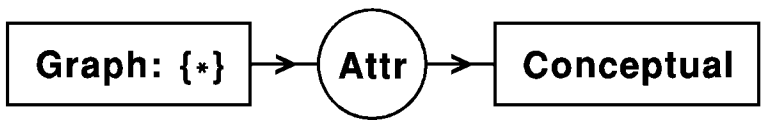

```
r = file (dir + "Factorial.CGF");
actor Factorial(n) is r.readGraph();
```

⇐ unnecessary



Some Future Directions

- Conjunction of preconditions in a rule should be treated as a goal. Internal backtracking required, e.g. for Sisyphus-I constraint re: room sharing and project members.
- Process constraints for real truth maintenance.
- Conformity relation should be implemented.
- Process algorithm needs to be reconciled with the CG Rules of Inference (Sowa, 1999); incorporate into pCG.
- Aspects of process algorithm bear re-examination, e.g. stopping criteria, graph morphology, π efficiency.
- Environment: IDE, Emacs mode.
- Expts/Apps: invoking a process within a process, temporal logic, business processes.
- Comparison against CLIPS, other CG mechanisms.



References

- **Delugach, H., Dynamic Assertion and Retraction of Conceptual Graphs, *Proceedings of the 6th Annual Workshop on Conceptual Graphs*, Eileen C. Way, editor, SUNY Binghampton, New York, pp. 15-26, July 1991.**
- **Linster, M., Documentation for the Sisyphus-I Room Allocation Task, [Accessed Online: November 2000], URL: http://ksi.cpsc.ucalgary.ca/KAW/Sisyphus/Sisyphus1/**
- **Mineau, G., From Actors to Processes: The Representation of Dynamic Knowledge Using Conceptual Graphs. In *Proceedings of the 6th International Conference on Conceptual Structures*, Montpellier, France, pp. 65–79, August 1998.**
- **Sowa, J.F., Conceptual Structures: Information Processing in Mind and Machine, Addison-Wesley, 1984.**
- **Sowa, J.F. et al, Conceptual Graph Standard, draft proposed American National Standard (dpANS) NCITS.T2/98-003, 1999. [Accessed Online: November 1999], URL: http://www.bestweb.net/~sowa/cg/cgdpansw.htm**