Computation and Control

Jason Hickey
Richard Murray

Caltech

Eric Klavins

Adam Granicz
Cristian Tapus
Justin Smith
Xin Yu
Nathan Gray

Computation and reaction

- A reactive system is a system that describes how to react to events
  - *"moving left" → "steer right"*
  - *"slowing down" → "increase thrust"*
- Algorithms are a set of rules to apply repeatedly
  - *while(i != 0) { k *= i; i-- }*

Rewriting systems

- A *rewriting* system us a language $\mathcal{L}$, together with a set of rules $s_i \rightarrow t_i$ for some $s_i, t_i \in \mathcal{L}$.
- s_i is called a *redex* (a pattern or set of terms)
- t_i is called a *contractum*
- A *computation* is a sequence of rewrite applications $e_1 \rightarrow e_2 \rightarrow \cdots \rightarrow e_n$

Lambda calculus (canonical example)

Language $\mathcal{L}$:

$$
\begin{array}{lcll}
e & ::= & v & \text{variables} \\
 & | & e_1\ e_2 & \text{function application} \\
 & | & \lambda v.e & \text{function abstraction}
\end{array}
$$

Rewrites:

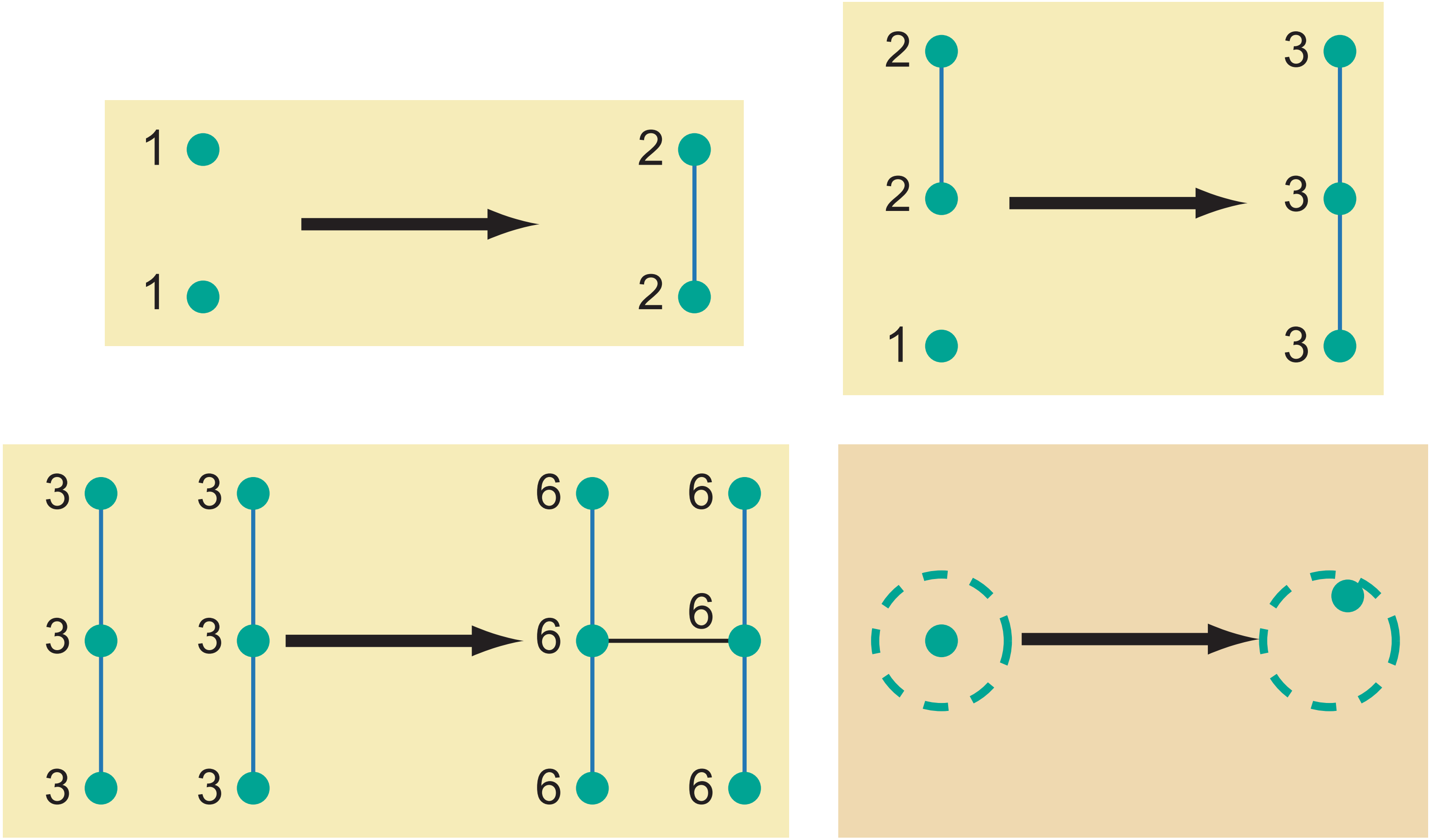
$$(\lambda v.e_1)\ e_2 \rightarrow_\beta e_1[e_2/v]$$

$$
\begin{array}{ll}
 & (\lambda x.\lambda y.x + y)\ 1\ 2 \\
\rightarrow & (\lambda y.1 + y)\ 2 \\
\rightarrow & 1 + 2 \\
\rightarrow & 3
\end{array}
$$

Example: vehicle formation/assembly

- Assemble a set of vehicle/parts into a formation

Formation rules

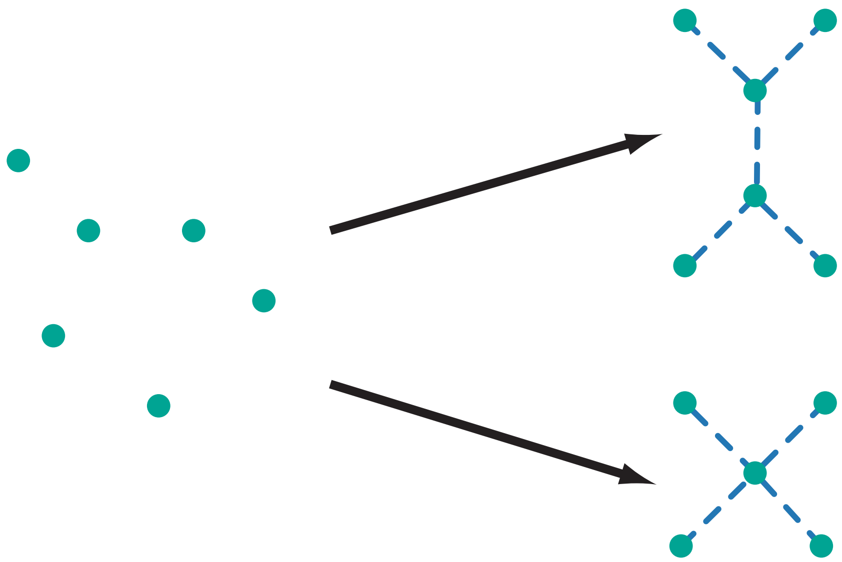


Rewriting rules

- Rule classes
  - *Progress: getting closer to a goal*
  - *Dynamics*
  - *Adversaries: destructive actions*
- Technical questions
  - *Determinism (Church-Rosser)*
  - *Progress (liveness)*
  - *Termination*
  - *Locality*

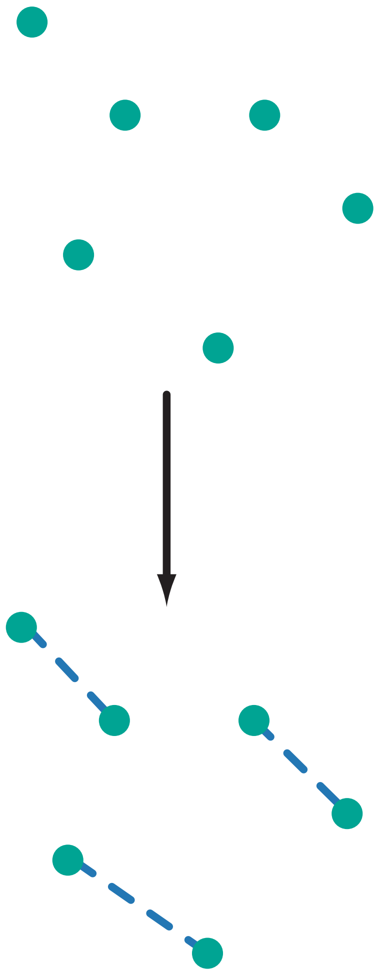
Determinism (Church-Rosser)

- Does the order of evaluation matter?
- Often the answer is no
  - *There may be reasons to have more than one result*
- Proofs are quite difficult



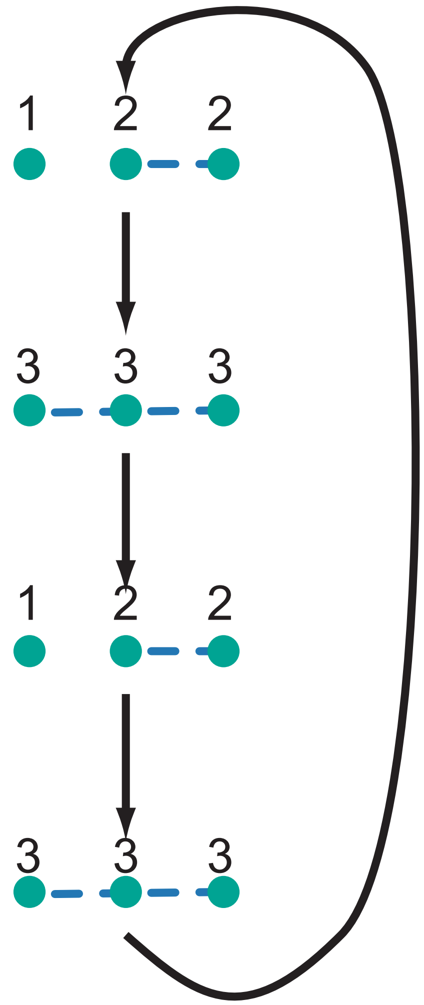
Deadlock/progress

- Is it possible to build a partial, final, formation?
  - *Show that: if a formation is not final, at least one rule is always enabled*
- Easy to prevent
  - *Add more rules to make progress from partial formations*
  - *Add "undo" to reverse bogus computations*



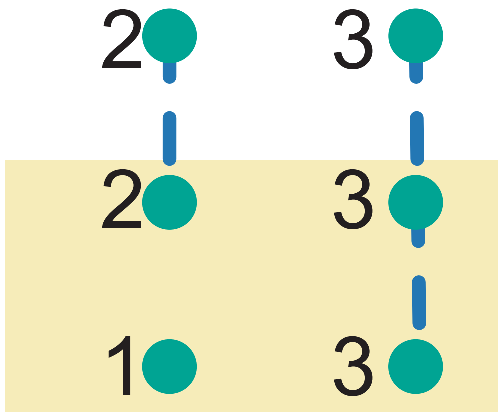
Termination

- Does every reduction sequence terminate?
- Termination proofs require the formation of a metric
  - *Often infeasible, if not it can be extremely hard*
  - *In many cases it doesn't matter*



Locality

- Is it possible to make decisions locally?
  - *Locality is determined by the scope of the rewrite*
- Methods
  - *Optimize the program to limit the scope*
  - *Introduce communication*



Languages

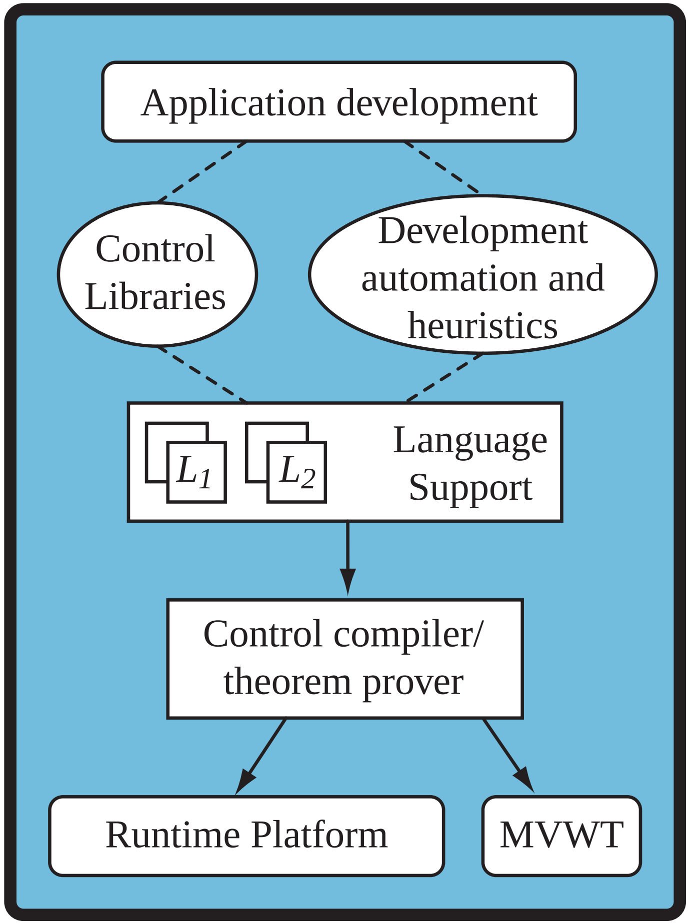
- Rewriting languages
  - *Determinism(*)*
  - *Termination(*)*
  - *Progress*
  - *Locality*

  - *(*) hard to prove, not always useful*

UNITY-like languages:

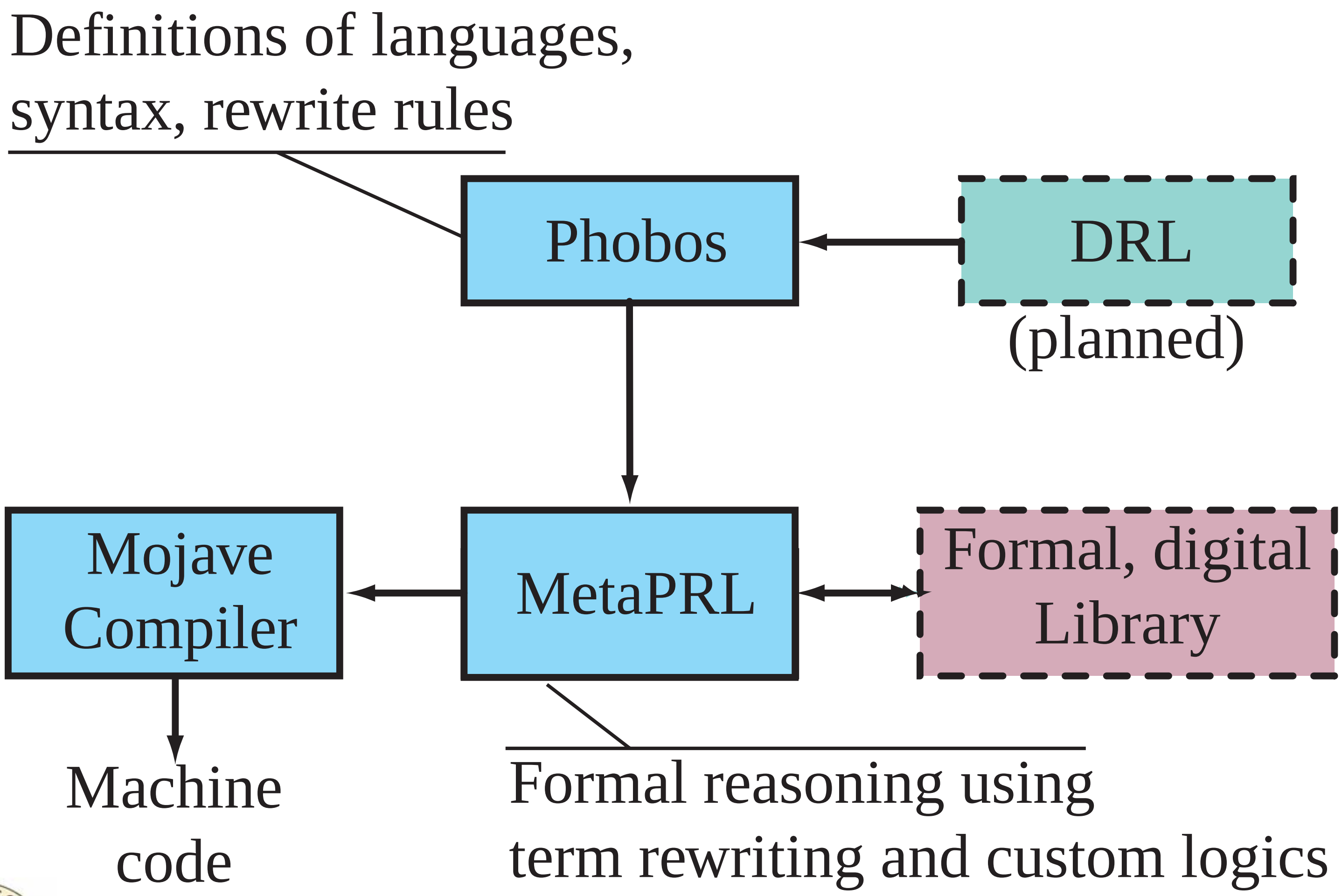
$$
\begin{array}{ccll}
G_1 & \rightarrow & P_1 & \text{decisions and actions} \\
G_2 & \rightarrow & P_2 & \\
 & \vdots & & \\
G_n & \rightarrow & P_n & \\
\hline
G_c & \rightarrow & P_c & \text{physics, control} \\
G_a & \rightarrow & P_a & \text{adversaries}
\end{array}
$$

Logical Programming Environments



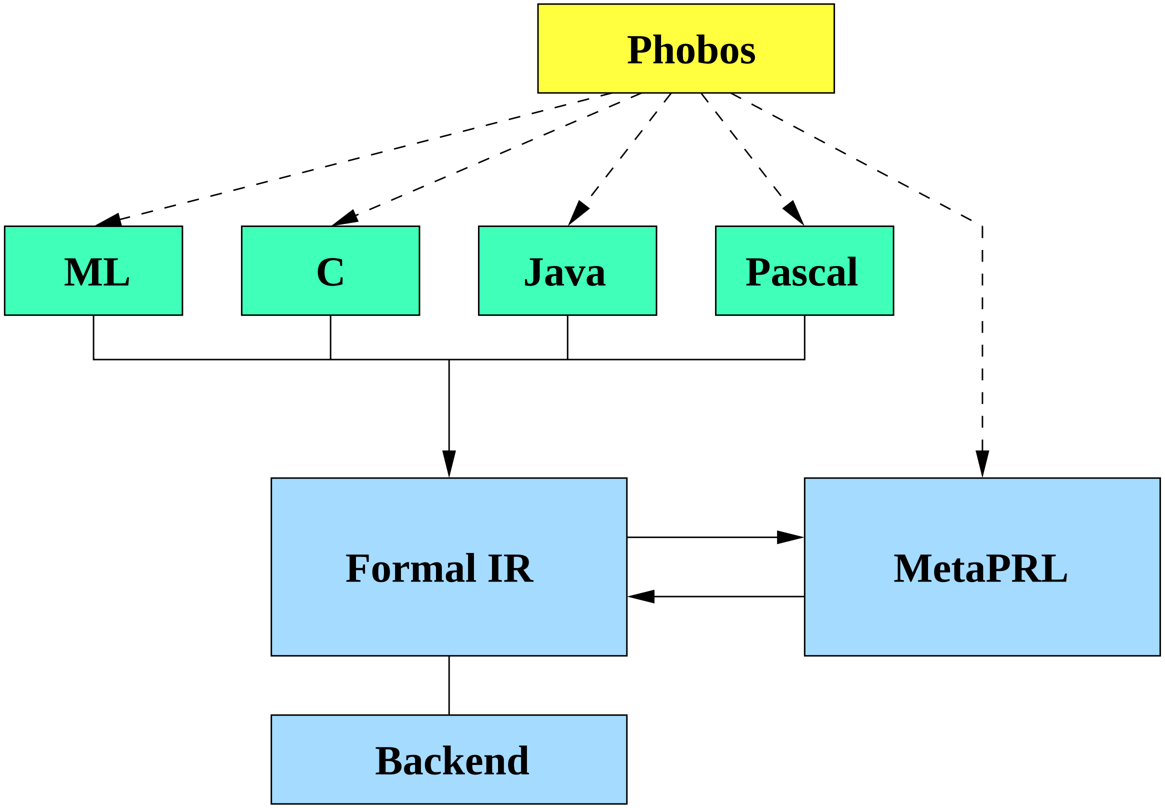
- The LPE is a framework for supporting formal design
  - Type theory *is a common language for specification and synthesis*
  - *Enables* collaborative *development of verified control libraries and design automation tools*
  - *The* compiler *is an assistant, and the link to executable code*

Logical Programming Environments



Phobos

- Phobos is a front-end for domain-specific languages
  - *Programs are translated to one of a set of "standard" languages*
  - *or to a theorem prover*



Language definitions

- Each language has a lexicon

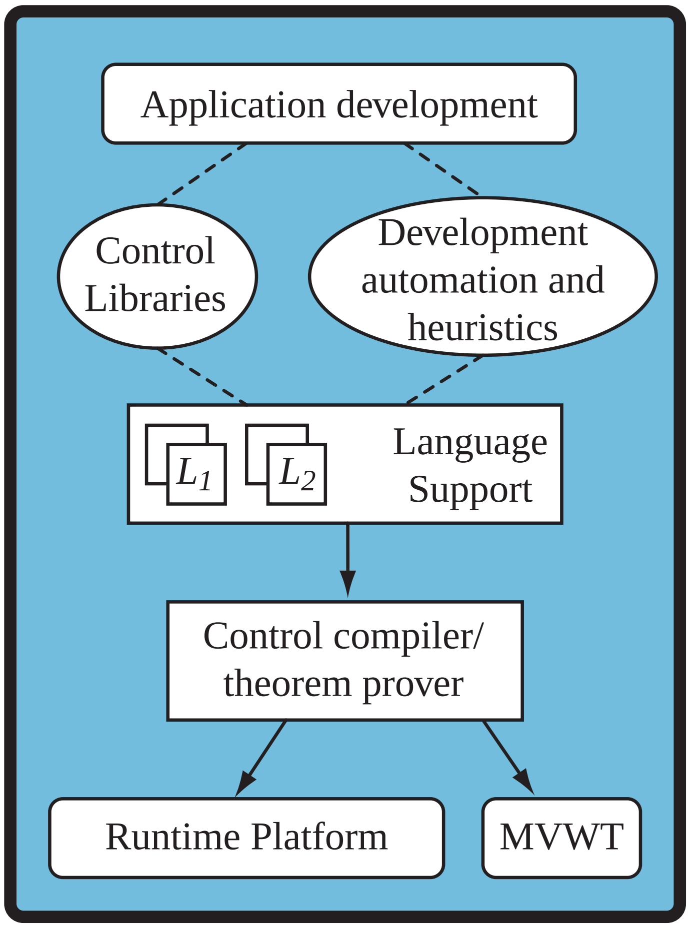
```
Tokens -longest {
   NUM = "[0-9]+"       { __token__[p:s]{'pos} -> num[p:s]{'pos} }
   ...
   * SPACE = " "        {}
}
```

- And a grammar

```
%left PLUS MINUS
%left TIMES DIV
%left LPAREN RPAREN

Grammar -start exp {
   exp ::= NUM          { num[p:s]{'pos} -> exp{num[p:s]; 'pos}}
      | ID              { ... }
      | exp PLUS exp    { 'e1 PLUS 'e2   -> exp{sum{'e1; 'e2};
                                               union_exp_pos{'e1; 'e2} }
      ...
}
```

Logical Programming Environments



- Parts
  - *Phobos language support complete*
  - *DRL language design and control primitives*
  - *Code compiler complete*