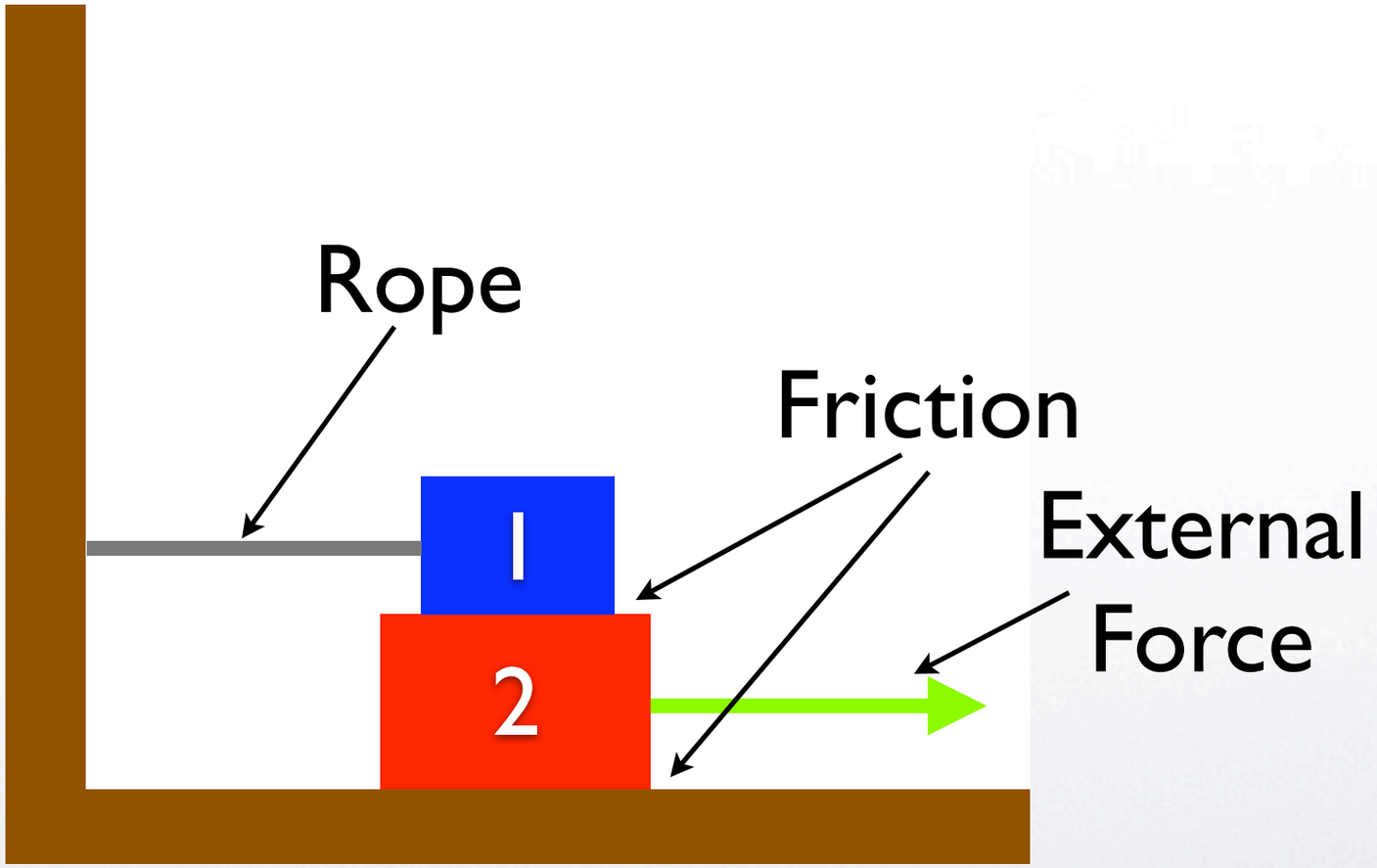
Solving Dynamic Non-Causal Equation Sets

Andrew Casey

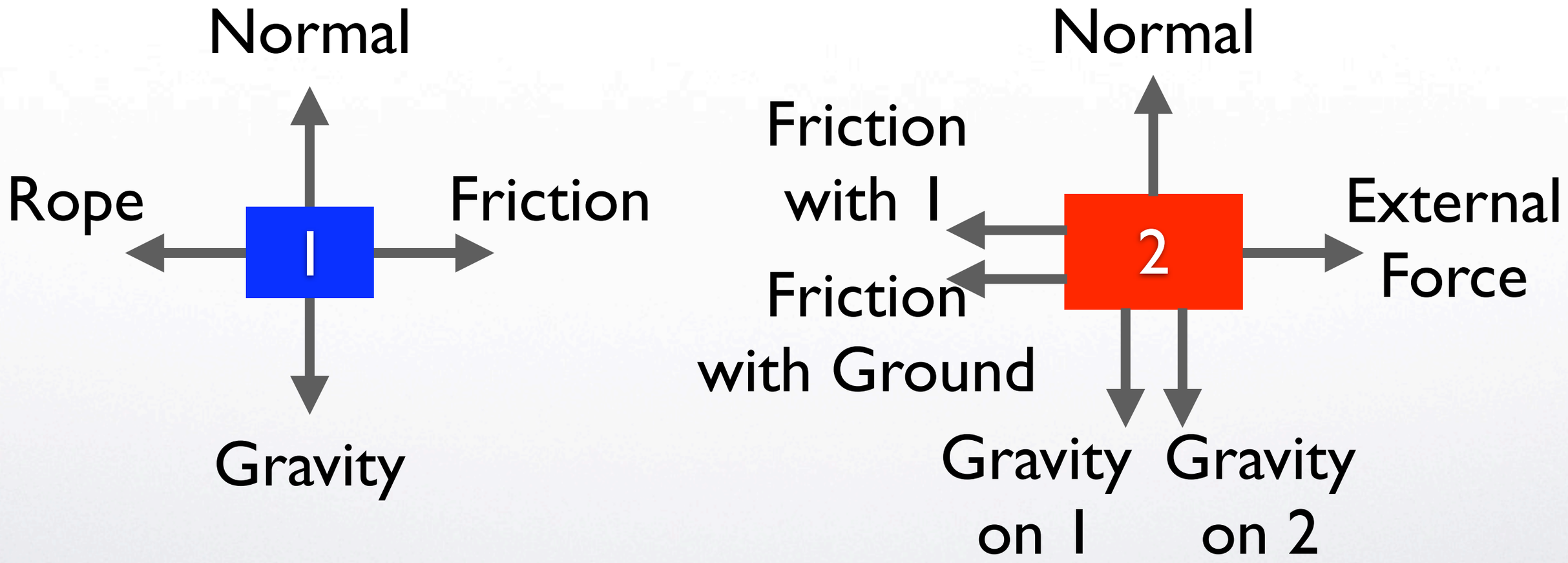
Review

A *non-causal equation* specifies mathematical equality rather than assignment

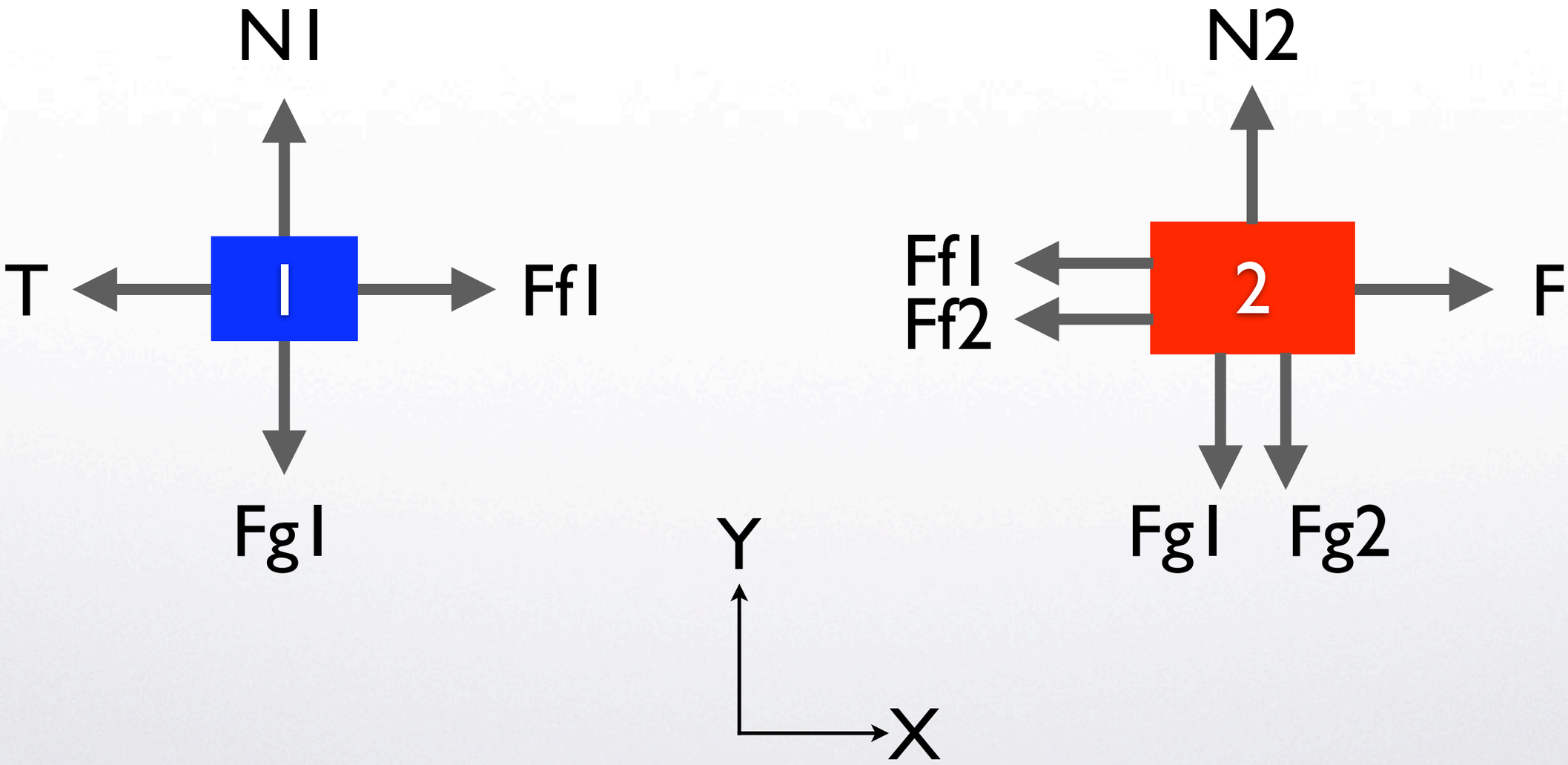
Motivation



Motivation


Normal
Rope
Friction
1
Gravity
Friction with 1
Normal
External Force
2
Friction with Ground
Gravity on 1
Gravity on 2

Motivation



Motivation

$F_{x1} = m_1 * A_{x1};$
$F_{y1} = m_1 * A_{y1};$

$F_{x1} = F_{f1} - T;$

$F_{y1} = N_1 - F_{g1};$

$F_{f1} = \mu * N_1;$

$F_{g1} = m_1 * g;$

$F_{x2} = m_2 * A_{x2};$
$F_{y2} = m_2 * A_{y2};$

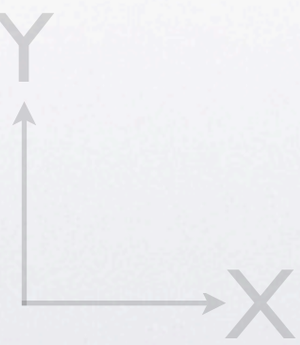
$F_{x2} = F - F_{f1} - F_{f2};$

$F_{y2} = N_2 - F_{g1} - F_{g2};$

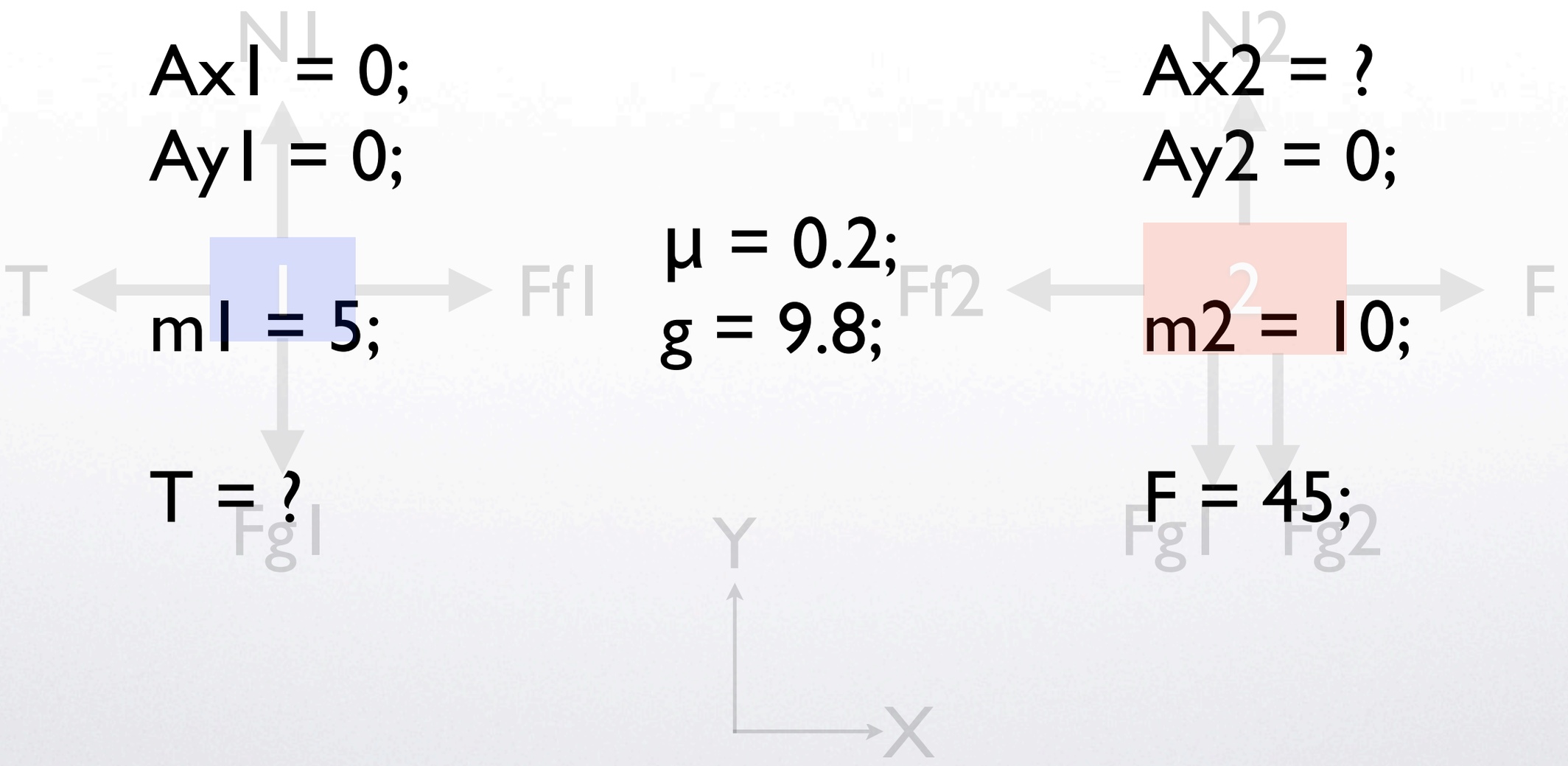
$F_{f2} = \mu * N_2;$

F_{g1} F_{g2}

g



Motivation



Ax1 = 0;
Ay1 = 0;

m1 = 5;

T = ?

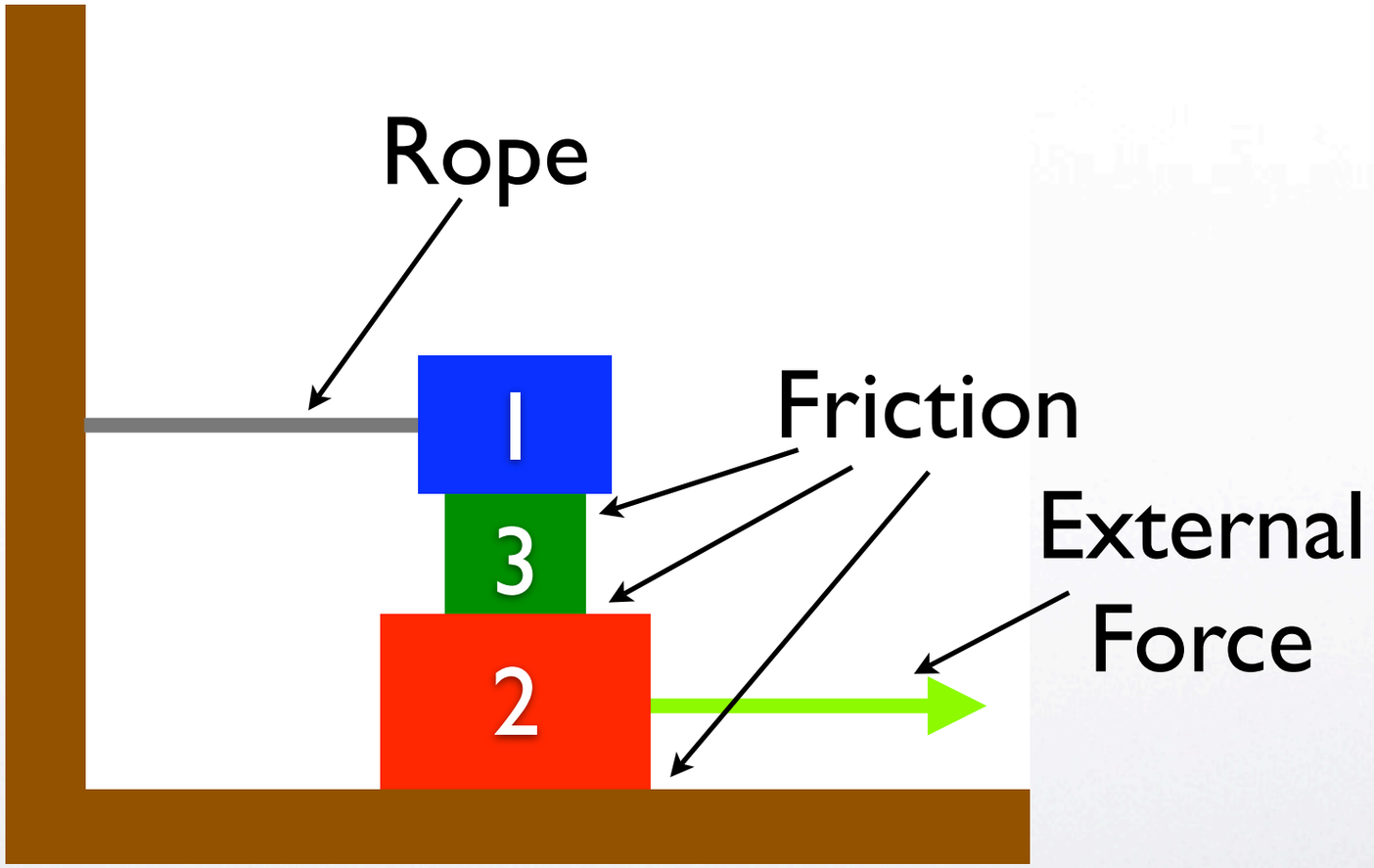
μ = 0.2;
g = 9.8;

Ax2 = ?
Ay2 = 0;

m2 = 10;

F = 45;

Motivation



Motivation

$F_{x1} = m_1 * A_{x1}$; $F_{x2} = m_2 * A_{x2}$;

$F_{y1} = m_1 * A_{y1}$; $F_{y2} = m_2 * A_{y2}$;

$F_{x1} = F_{f1} - T$; $F_{x2} = F - F_{f2} - F_{f1}$;

$F_{y1} = N_1 - F_{g1}$; $F_{y2} = N_2 - F_{g1} - F_{g2}$;

$F_{f1} = \mu * N_1$; $F_{f2} = \mu * N_2$;

$F_{g1} = m_1 * g$; $F_{g2} = m_2 * g$;

Motivation

```
Fx1 = m1 * Ax1;    Fx2 = m2 * Ax2;                    Fx3 = m3 * Ax3;
Fy1 = m1 * Ay1;    Fy2 = m2 * Ay2;                    Fy3 = m3 * Ay3;

Fx1 = Ff1 - T;     Fx2 = F - Ff2 - Ff3;               Fx3 = Ff3 - Ff1;
Fy1 = N1 - Fg1;    Fy2 = N2 - Fg1 - Fg2 - Fg3;        Fy3 = N3 - Fg1 - Fg3;

Ff1 = μ * N1;      Ff2 = μ * N2;                      Ff3 = μ * N3;

Fg1 = m1 * g;      Fg2 = m2 * g;                      Fg3 = m3 * g;
```

Process

- Scanning & Parsing
- Simplification
- Causality Assignment
- Equation Ordering
- Evaluation

Scanning & Parsing

- SableCC for the CST to the AST transformation and the generated tree walkers

Simplification

- OpSimplifier + BinOpFlattener & ConstantFolder
- Until fixed point:
  - ExponentDistributor + BinOpFlattener & ConstantFolder
  - ExponentGatherer + BinOpFlattener & ConstantFolder
  - LiteralDistributor +BinOpFlattener & ConstantFolder
- CanonicalSorter
- TermGatherer + BinOpFlattener & ConstantFolder
- CanonicalSorter

Causality Assignment

- Dinic's algorithm for maximum network flow
- Only consider solvable variables

Equation Ordering

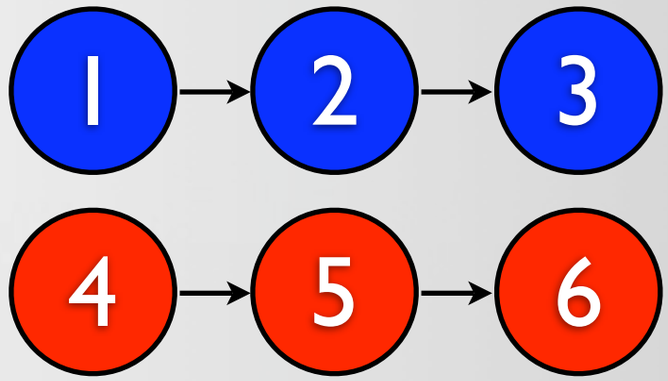
- Build an equation dependency graph
- Find the strongly-connected components (SCCs)
- Build a DAG of SCCs
- Sneaky bit: missing variables

Evaluation

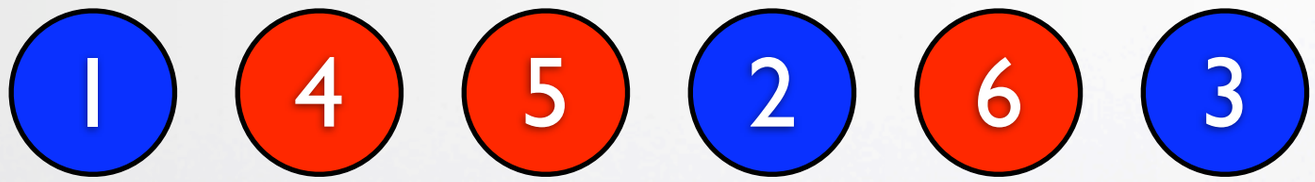
- *Single-threaded*: evaluate the SCCs in topological order
- *Naive Multi-threaded*: one thread per SCC
- *Pooled Multi-threaded*: use a fixed size pool of available threads and attempt to assign child SCCs to threads already processing parent SCCs

Example

DAG

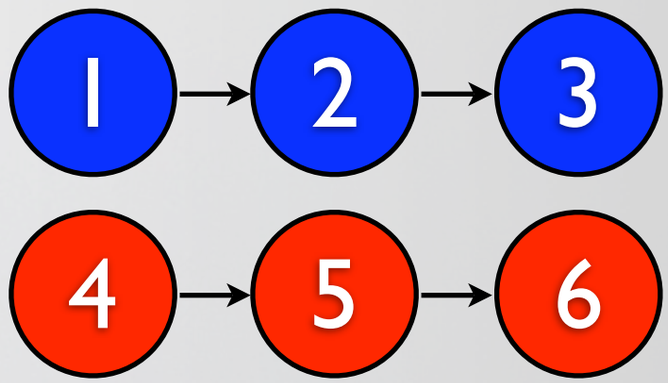


A Topological Ordering



Thread 1

Thread 2



Solving

- *Numeric*: pass the whole thing to Octave
- *Naive*: mixed symbolic/numeric approach already described
- *Incremental*: make use of existing data structures rather than starting from scratch

Incremental Addition

- Always update the list of equations, the variable frequency counts, and the solvable variable map

Incremental Addition

- Important Cases:
  - If the old system is inconsistent, then solve all
  - If the new system doesn't use any variables from the old system, solve separately and merge
  - If one of the systems is much bigger, solve it first and try to merge

Incremental Deletion

- Always update the list of equations, the variable frequency counts, and the solvable variable map

Incremental Deletion

- Important Cases:
  - If the old system is inconsistent, then solve all
  - If most of the old system is deleted, then solve all
  - If nothing depends on the deleted equations, then just delete them (and clean up the data structures)

Incremental Change

- Delete and then Add
- Special Case: X = C
- Favour predictability over marginal improvement

Results

One Thread	Base	Extended
Numeric[1]	1036 ms	2151 ms
Naive	501 ms	1879 ms
Incremental (1.0)	502 ms	540 ms
Incremental (0.5)	483 ms	1796 ms

Two Threads	Base	Extended
Numeric[1]	1062 ms	2128 ms
Naive	495 ms	1849 ms
Incremental (1.0)	499 ms	593 ms

1) Input size was 1/25

References

- Cormen, Thomas H., Leiserson, Charles E., Rivest, Ronald L., Stein, Clifford: Introduction to Algorithms, Second Edition. Sect. 22.5. MIT Press, Cambridge, MA (2001)
- Dinic, E.A.: Algorithm for solution of a problem of maximum flow in a network with power estimation. Soviet Math. Dokl., 11, 1277–1280 (1970)
- Indrani, A.V.: Some issues concerning Computer Algebra in AToM^3. Technical Report, School of Computer Science, McGill University, Montreal, QC (2003)
- Vangheluwe, Hans, Sridharan, Bhama, Indrani, A.V.: An algorithm to implement a canonical representation of algebraic expressions and equations in AToM^3. Technical Report, School of Computer Science, McGill University, Montreal, QC (2003)
- Xu, Weigao "Steven": The Design and Implementation of the μModelica compiler. MSc. Thesis. School of Computer Science, McGill University, Montreal, QC (2005)

Demo