

UNITY Logic

(Based on the Modified Version in [Misra 1995])

Yih-Kuen Tsay

Department of Information Management
National Taiwan University

Introduction

UNITY was once quite popular. Its logic has been modified and improved in a subsequent work.

J. Misra. A logic for concurrent programming. Journal of Computer and Software Engineering, 3(2): 239-272, 1995.

- 🌐 A program consists of (1) an **initial condition** and (2) a **set of actions** (or conditional multiple-assignments), which always includes *skip*.
- 🌐 Properties are defined in terms of
 - ☀ initially p ,
 - ☀ p **co** q , and
 - ☀ p transient.

Program Model: Action System

- 🌐 Syntax: An action system consists of
 - ☀️ a set of *variables* and
 - ☀️ a set of *actions*, always including *skip* (which does not change the system's state).

A particular valuation of the variables is called a system or program *state*. An action is essentially a *guarded multiple assignment* to the variables.

- 🌐 Semantics:
 - ☀️ A system execution starts from some initial state and goes on forever.
 - ☀️ In each step of an execution, some action is selected (under some fairness constraint) and executed, resulting in a possible change of the program state.

The “Constrains” Operator

- 🌐 The safety properties of a system are stated using the “constrains” (**co**) operator.
- 🌐 “ $p \text{ co } q$ ” (p constrains q) states that whenever p holds, q holds after the execution of any single action.
- 🌐 Formally, $p \text{ co } q \triangleq \langle \forall t :: \{p\} t \{q\} \rangle$.
- 🌐 As *skip* may be applied in any state, from $p \text{ co } q$ it follows that $p \Rightarrow q$.
- 🌐 It also follows that once p holds, q continues to hold upto (and including) the point where p ceases to hold (if it ever does).

Usages of the **co**

- “ $x = 0$ **co** $x \geq 0$ ”: once x becomes 0 it remains 0 until it becomes positive.
- “ $\forall m :: x = m$ **co** $x \geq m$ ”: x never decreases.
This is equivalent to “ $\forall m :: x \geq m$ **co** $x \geq m$ ”.
- “ $\forall m, n :: x, y = m, n$ **co** $x = m \vee y = n$ ”: x and y never change simultaneously.

The *unless* Operator


- “*p unless q*” was introduced in the original UNITY logic as a basic safety property:

$$p \text{ unless } q \text{ in } F \triangleq \forall t : t \text{ in } F : \{p \wedge \neg q\} t \{p \vee q\}$$

If *p* is *true* at some point of computation, then it will continue to hold as long as *q* does not (*q* may never hold and *p* continues to hold forever).

- Example: “ $x \geq k$ *unless* $x > k$ ” says that *x* is non-decreasing.
- $p \text{ unless } q \equiv p \wedge \neg q \text{ **co** } p \vee q.$
- $p \text{ **co** } q \equiv p \text{ unless } \neg p \wedge q.$

Special Cases of co

 p stable $\triangleq p \text{ co } p$

 p invariant $\triangleq (\text{initially } p) \text{ and } (p \text{ stable})$

Some Rules of Hoare Logic

$$\frac{}{\{p\} s \{true\}} \qquad \frac{}{\{false\} s \{q\}}$$

$$\frac{\{p\} s \{false\}}{\neg p}$$

$$\frac{\langle \forall j :: \{p_j\} s \{q_j\} \rangle}{\{\langle \forall j :: p_j \rangle\} s \{\langle \forall j :: q_j \rangle\}} \qquad \frac{\langle \forall j :: \{p_j\} s \{q_j\} \rangle}{\{\langle \exists j :: p_j \rangle\} s \{\langle \exists j :: q_j \rangle\}}$$

$$\frac{p \Rightarrow p', \{p'\} s \{q'\}, q' \Rightarrow q}{\{p\} s \{q\}}$$

Derived Rules (Theorems)

A theorem in the form of

$$\frac{\Delta_1}{\Delta_2}$$

means that properties in Δ_2 can be deduced from properties in the premise Δ_1 .

Some Derived Rules

🌐 *false* **co** *p*.

🌐 *p* **co** *true*.

🌐 Conjunction and Disjunction

$$\frac{p \text{ co } q, p' \text{ co } q'}{p \vee p' \text{ co } q \vee q'}$$

$$p \wedge p' \text{ co } q \wedge q'$$

🌐 Stable Conjunction and Disjunction

$$\frac{p \text{ co } q, r \text{ stable}}{p \wedge r \text{ co } q \wedge r}$$

$$p \vee r \text{ co } q \vee r$$

The Substitution Axiom

An invariant may be replaced by *true*, and vice versa, in any property of a program.

🌐 Example 1: given $p \text{ co } q$ and J invariant, we conclude

$$p \wedge J \text{ co } q, p \text{ co } q \wedge J, p \wedge J \text{ co } q \wedge J, \text{ etc.}$$

🌐 Example 2:

$$\frac{p \text{ unless } q, \neg q \text{ invariant}}{p \text{ stable}}$$

Note: there is a distinction between an invariant (of a particular program) and a valid formula (in any context). However, as part of a program property, they can be safely interchanged.

An Elimination Theorem

- Free variables may be eliminated by taking conjunctions or disjunctions.
- Suppose p a property that does not name any program variable other than x .
- Then, $p[x := m]$ does not contain any variable and is a constant (and hence stable).
- Observe that $p = \langle \exists m : p[x := m] : x = m \rangle$.
- An elimination theorem:

$$\frac{x = m \text{ co } q, \text{ where } m \text{ is free} \\ p \text{ does not name } m \text{ nor any program variable other than } x}{p \text{ co } \langle \exists m :: p[x := m] \wedge q \rangle}$$

An Elimination Theorem (cont.)

$$\frac{x = m \text{ co } q, \text{ where } m \text{ is free} \\ p \text{ does not name } m \text{ nor any program variable other than } x}{p \text{ co } \langle \exists m :: p[x := m] \wedge q \rangle}$$

Proof:

$$\begin{aligned} x = m \text{ co } q & \quad , \text{ premise} \\ p[x := m] \wedge x = m \text{ co } p[x := m] \wedge q & \quad , \text{ stable disjunction with } p[x := m] \\ \langle \exists m :: p[x := m] \wedge x = m \rangle \text{ co } \langle \exists m :: p[x := m] \wedge q \rangle & \quad , \text{ disjunction over all } m \\ p \text{ co } \langle \exists m :: p[x := m] \wedge q \rangle & \quad , \text{ simplifying the lhs} \end{aligned}$$

Transient Predicate (under Weak Fairness)

- Under weak fairness, it is sufficient to have a single action falsify a transient predicate.
- p transient $\stackrel{\Delta}{\equiv} \langle \exists s :: \{p\} s \{\neg p\} \rangle$
- Some derived rules:

$$(p \text{ stable} \wedge p \text{ transient}) \equiv \neg p$$

(The only predicate that is both stable and transient is *false*.)

$$\frac{p \text{ transient}}{p \wedge q \text{ transient}}$$

Progress Properties

- 🌐 p ensures $q \triangleq (p \wedge \neg q \text{ co } p \vee q)$ and $p \wedge \neg q$ transient.
If p holds at any point, it will continue to hold as long as q does not hold; eventually q holds.
- 🌐 “ $p \mapsto q$ ” specifies that if p holds at any point then q holds or will eventually hold. Inductive definition:

$$\frac{p \text{ ensures } q}{p \mapsto q}$$

$$\text{(transitivity)} \quad \frac{p \mapsto q, q \mapsto r}{p \mapsto r}$$

$$\text{(disjunction)} \quad \frac{\langle \forall m : m \in W : p(m) \mapsto q \rangle}{\langle \exists m : m \in W : p(m) \rangle \mapsto q}$$

Example: “ $x \geq k \mapsto x > k$ ” says that x will eventually increase.


Some Derived Rules for Progress

🌐 (Progress-Safety-Progress, PSP)




$$\frac{p \mapsto q, r \text{ co } s}{(p \wedge r) \mapsto (q \wedge s) \vee (\neg r \wedge s)}$$

🌐 (well-founded induction)

$$\frac{\langle \forall m :: p \wedge M = m \mapsto (p \wedge M < m) \vee q \rangle}{p \mapsto q}$$

 Notation: $F \parallel G$ (the *union* of F and G)

 Semantics:

-  The set of variables is the *union* of the two sets of variables.
-  The set of actions is the *union* of the two sets of actions.
-  The composed system is executed as a single system.

UNITY Logic vs. Lamport's 'Hoare Logic'

- “**co**” enjoys the complete rule of consequence.
- Rules of conjunction and disjunction also hold.
- Stronger rule of parallel composition:

$$\frac{p \text{ co } q \text{ in } F, p \text{ co } q \text{ in } G}{p \text{ co } q \text{ in } F \parallel G}$$

- But, “**co**” is much less convenient for sequential composition.

Union Theorems

$$\frac{\text{🌐 } p \text{ unless } q \text{ in } F, p \text{ stable in } G}{p \text{ unless } q \text{ in } F \parallel G}$$

$$\frac{\text{🌐 } p \text{ invariant in } F, p \text{ stable in } G}{p \text{ invariant in } F \parallel G}$$

$$\frac{\text{🌐 } p \text{ ensures } q \text{ in } F, p \text{ stable in } G}{p \text{ ensures } q \text{ in } F \parallel G}$$

- 🌐 If any of the following properties holds in F , where p is a local predicate of F , then it also holds in $F \parallel G$ for any G :
 p unless q , p ensures q , p invariant.

Note: Any invariant used in applying the substitution axiom to deduce a property of one module should be proved an invariant in the other module.