Conjoining Specifications I

Wolfgang Schreiner
Research Institute for Symbolic Computation
Johannes Kepler University, Linz, Austria

Wolfgang.Schreiner@risc.uni-linz.ac.at
http://www.risc.uni-linz.ac.at/people/schreine

Parallel Program Composition

- Large systems built from small parts.
  - Component π_i
  - Formula S_i
- Deduce system properties from components
  - **cobegin** $\pi_1 \parallel \ldots \parallel \pi_n$
  - $\mathsf{S}_1 \wedge \ldots \wedge \mathsf{S}_n$
- Composition is conjunction.
  - Best way for proving properties of composite systems.
- Use TLA as the logical basis
  - Specification of components.
  - Specification of environment.
- Composite specifications arise from:
  - *Decompose* system into smaller parts.
  - *Compose* parts to form larger system.

Composed Specifications

- Decomposition:
  - Environment of each component are the other components.
  - Assumption about environment must be stated.
  - Assumption must be satisfied by other components.
  - *Decomposition theorem* for system verification by proving property of high-level specification and properties of each low-level component.
- Composition:
  - Environment is unknown.
  - Component satisfies guarantee M only under assumption E.
  - Assumption/guarantuee property $M \overset{+}{\rhd} E$.
  - *Composition theorem* for reasoning about assumption/guarantee specifications.

Start with informal presentation.

Decomposing Complete Systems

- Complete System.
  - Self-contained system that may be observed but does not interact with observer.
- Example: GCD
  - Program GCD
    var a **initially** 233344, B **initially** 233577899;
    cobegin
    loop $\langle$ **if** $a > b$ **then** $a := a - b$ $\rangle$ **endloop**
    $\|$
    loop $\langle$ **if** $b > a$ **then** $b := b - a$ $\rangle$ **endloop**
    coend
- Correctness property:
  - a and b eventually become and remain equal to gcd of initial values.
  - Program M_{gcd}, correctness property P_{gcd}
  - Verification $M_{gcd} \Rightarrow P_{gcd}$

Hierarchical Development

- Decomposition of specification:
  - $M_{gcd} = M_a \wedge M_b$
  - M_a asserts that a is initially 233344 and repeatedly decremented by b whenever $a > b$.
- Decomposition of program:
  - $\pi = \pi_a \parallel \pi_b$
  - Process π_a

    output var a **initially** 233344;
    input var b;
    loop $\langle$ **if** $a > b$ **then** $a := a - b$ $\rangle$
    endloop
  - Process π_b

    output var b **initially** 233577899;
    input var a;
    loop $\langle$ **if** $b > a$ **then** $b := b - a$ $\rangle$
    endloop

Refinement of Components

- By decomposition, components can be refined separately:
  - Process π_a^l
    output var a **initially** 233344;
    internal var ai;
    input var b;
    loop
    $\langle\ ai := b\ \rangle$;
    if $\langle\ a > ai\ \rangle$ **then** $\langle\ a := a - ai\ \rangle$
    endloop
- Correctness property:
  - Compositions of refined components implements GCD.
  - $M_a^l \wedge M_b^l \Rightarrow M_a \wedge M_b$
- Decomposition of proofs not so easy:
  - $\pi_a^l \Rightarrow \pi_a$ does *not* hold!

Decomposition of Proofs

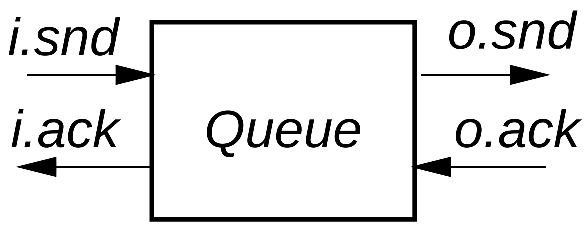
- π_a^l does not implement π_a
  - No assumptions when b can change.
  - π_a^l can behave in ways that π_a cannot!
  - π_a can increment a only by current value of b.
  - π_a^l can increment a by previous value of b (if b changes after assignment to ai)
- Context assumption:
  - $E_a \equiv b$ does not change when $a > b$.
  - $E_a \wedge M_a^l \Rightarrow M_a$.
- Basic proof idea (not yet fully correct)
  - $E_a \wedge M_a^l \Rightarrow M_a$
    $E_b \wedge M_b^l \Rightarrow M_b$
    $M_a \wedge M_b \Rightarrow E_a \wedge E_b$
- Decomposition leads simpler proofs:
  - E_a significantly simpler than M_b^l.
  - $M_a \wedge M_b \Rightarrow E_a \wedge E_b$ simpler than direct proof of specification.

Composing Open Systems

- Open System:
  - Interaction with uncontrolled environment.
- Example: two-phase hand-shake protocol
  - Channel $c = \langle\ c.val,\ c.sig,\ c.ack\ \rangle$
  - Message value $c.val$
  - Synchronization bits $c.snd = \langle\ c.sig,\ c.ack\ \rangle$
  - Channel ready to send when $c.sig = c.ack$
  - Sending v by setting $c.val$ to v and complementing $c.sig$.
  - Recept acknowledged by complementing $c.ack$
- State sequence for sending $37, 4, 19, \ldots$

$c.ack$	0	0	1	1	0	0	...
$c.sig$	0	1	1	0	0	1	...
$c.val$	-	37	37	4	4	19	...

A Queue Process

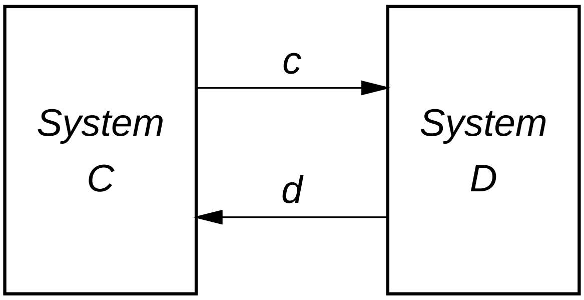


Process Queue
output var $i.ack$, $o.sig$ **initially** 0, $o.val$;
internal var q **initially** $\langle q \rangle$;
input var $i.sig$, $i.val$, $o.ack$;
cobegin
  loop $\langle$
  if $(i.ack \neq i.sig) \wedge (|q| < N)$
  then $q := q \circ \langle i.val \rangle$; $i.ack := \neg i.ack$
  $\rangle$ **endloop**
$\|$
  loop $\langle$
  if $(o.ack \neq o.sig) \wedge (|q| > 0)$
  then $o.val := head(q)$; $q := tail(q)$; $o.sig := \neg o.sig$
  $\rangle$ **endloop**
coend

Queue Specification

- Specification of queue:
  - TLA formula QM representing queue process.
  - QM is not specification of queue.
  - Environment might not obey communication protocol!
- Constrained specification:
  - Need not rule out bad queue behavior if environment performs incorrectly.
  - Assumption/guarantuee style of specification.
  - Assumption E asserts that environment obeys protocol.

Reasoning about Composed Systems



- Composition of systems C and D
  - C guarantees M_c assuming M_d.
  - D guarantees M_d assuming M_c.
  - Can we conclude that composition implement $M_c \wedge M_d$ with no environment assumption?
- Implementation:
  - Process π_c
    output var c **initially** 0;
    input var d;
    loop $\langle\ c := d\ \rangle$ **endloop**
  - Process π_d
    output var d **initially** 0;
    input var c;
    loop $\langle\ d := c\ \rangle$ **endloop**

Reasoning about Composed Systems

- Example:
  - M_c^0 asserts c always equals 0.
  - M_d^0 asserts d always equals 0.
  - π_c guarantees M_c^0 assuming M_d^0 and π_d guarantees M_d^0 assuming M_c^0.
  - Composition leaves c and d unchanged, so it implements $M_c^0 \wedge M_d^0$.
- Example:
  - M_c^1 asserts c eventually equals 1.
  - M_d^1 asserts d eventually equals 1.
  - π_c guarantees M_c^0 assuming M_d^0 and π_d guarantees M_d^0 assuming M_c^0.
  - Composition leaves c and d unchanged, so it does *not* implement $M_c^1 \wedge M_d^1$!

Reasoning about Composed Systems

- First Example:
  - Changing the output variable would violate guarantee *before* assumption had been violated.
- Second Example:
  - Violating the quarantee does not occur at any particular moment in time.
- Assumption/guarantuee specifications:
  - Guarantuee M can become false only after assumption E becomes false.

Reasoning about composition of specification is easiest when assumptions are safety conditions!

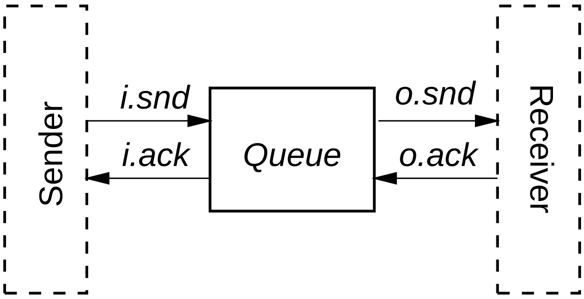
TLA

- Canconical TLA specification form:
  - $\exists x$: $Init \land \Box[\mathsf{N}]_v \land L$.
  - Initialization predicate $Init$.
  - Variable vector v.
  - Fairness conditions L.
  - There exists a sequence of values for x such that *Init* is true in the initial state, every step of the behavior is an N step or leaves state function v unchanged and L holds.
- Example: GCD
  - $Init \equiv (a = 233344) \land (b = 233577899)$
    $\mathsf{N} \equiv ((a > b) \land (a' = a - b) \land (b' = b))$
    $\quad \lor ((b > a) \land (b' = b - a) \land (a' = a))$
    $v \equiv \langle a, b \rangle$
    $L \equiv \mathsf{WF}_v(\mathsf{N})$

Interleaving Representations

- Let ξ and ψ be two objects.
  - Representation by variables x and y.
  - How to model concurrent changes to ξ and ψ?
  - Simultaneous change to x and y?
  - Separate changes to x and y in some order?
- *Interleaving representations*
  - Simultaneous changes to x, y are disallowed.
  - x, y may be output variables of different processes.
  - System with n components in which v_i is tuple of output variables of component i.
  - Interleaving represented by TLA formula
    $Disjoint(v_1, \ldots, v_n) \equiv$
    $\wedge_{i \neq j} \Box[(v_i{}' = v_i) \vee (v_j{}' = v_j)]_{\langle v_i, v_j \rangle}$
- TLA uses interleaving representation.
  - Adequate for *any* system that is modeled at sufficiently fine grain of atomicity.

The Queue Example



- Channel is initially ready for sending:
  - $CInit(c) \equiv (c.sig = c.ack = 0)$
- Sending a value v on channel c
  - $Send(v, c) \equiv c.sig = c.ack$
    $\land\ c.snd' = \langle 1 - c.sig, v \rangle$
    $\land\ c.ack' = c.ack$
- Acknowledging recept of value on c
  - $Ack(c) \equiv c.sig \neq c.ack$
    $\land\ c.ack' = 1 - c.ack$
    $\land\ c.snd' = c.snd$
- Representation as complete system
  - Add environment that sends arbitrary numbers over i and acknowledges values on o.

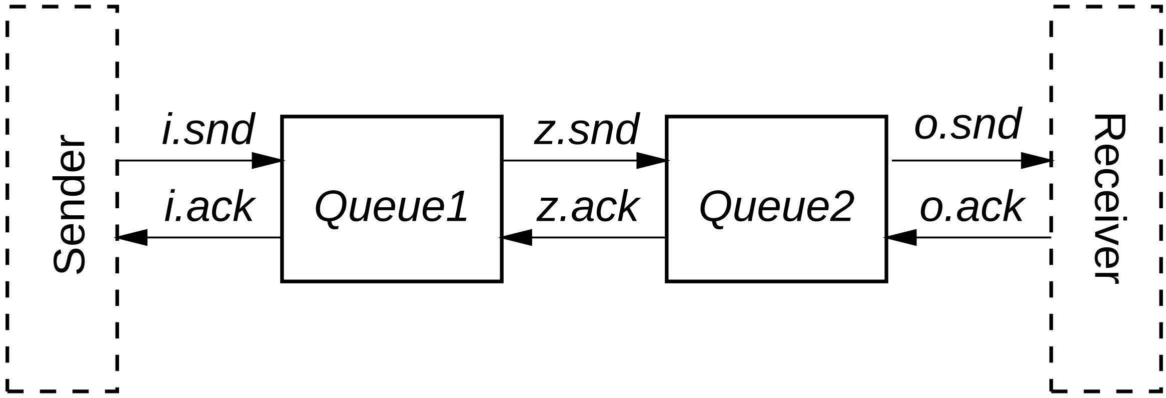
TLA Specification of Queue

- Environment actions
  - $Init_E \equiv CInit(i)$
  - $Put \equiv (\exists v \in \mathsf{Nat}\colon Send(v,i)) \wedge (o' = o)$
  - $Get \equiv Ack(o) \wedge (i' = i)$
  - $Q_E \equiv Get \vee Put$
- Component actions
  - $Init_M \equiv CInit(o) \wedge (q = \langle\rangle)$
  - $Enq \equiv |q| < N \wedge Ack(i)$
    $\wedge (q' = q \circ \langle i.val \rangle) \wedge (o' = o)$
  - $Deq \equiv |q| > 0 \wedge Send(Head(q), o)$
    $\wedge (q' = Tail(q)) \wedge (i' = i)$
  - $Q_M \equiv Enq \vee Deq$
- Complete System Specification
  - $ICL \equiv \mathsf{WF}_{\langle i,o,q \rangle}(Q_M)$
  - $ICQ \equiv Init_E \wedge Init_M$
    $\wedge \Box[(QE \wedge (q' = q)) \vee Q_M]_{\langle i,o,q \rangle} \wedge ICL$
  - $CQ \equiv \exists q\colon ICQ$

Implementation

- Specification M^l implements M.
  - $M^l \Rightarrow M$
  - Every behavior satisfying M^l satisfies M.
- Special form
  - $\exists x$: $\widehat{\mathsf{M}}$
  - Proof requires *refinement mapping*
  - Tuple of state functions $\overline{x}$ such that $M^l \Rightarrow \overline{\widehat{M}}$.
  - $\overline{\widehat{M}}$ is $\widehat{M}$ with $\overline{x}$ substituted for x.

Two Queues in Series



$$
\begin{aligned}
&ICDQ \equiv Init_E \wedge Init_M^{[1]} \wedge Init_M^{[2]} \\
&\wedge \Box \left[(Q_E \wedge \langle q_1,\, q_2,\, z\rangle' = \langle q_1,\, q_2,\, z\rangle) \right. \\
&\quad \vee (Q_M^{[1]} \wedge \langle q_2,\, o\rangle' = \langle q_2,\, o\rangle) \\
&\quad \left. \vee (Q_M^{[2]} \wedge \langle q_1,\, i\rangle' = \langle q_1,\, i\rangle) \right]_{\langle i, o, z, q_1, q_2\rangle} \\
&\wedge ICL^{[1]} \wedge ICL^{[2]} \\
&CDQ \equiv \exists q_1, q_2 \colon\ ICDQ
\end{aligned}
$$

Specification

- Example: two queues in series.
  - Implementation of singler larger queue.
  - $F^{[1]} \equiv F[z/o, q_1/q]$.
  - $F^{[2]} \equiv F[z/i, q_2/q]$
- Composite queue implements $(2N+1)$-queue
  - Channel z acts as additional buffer.
  - $F^{[dbl]} \equiv F[(2N+1)/N]$
  - $CDQ \Rightarrow CQ^{[dbl]}$.
- Proof idea:
  - $ICDQ \Rightarrow \overline{ICQ^{[dbl]}}$
  - Refinement mapping

    $q :=$ **if** $z.sig = z.ack$

    then $q_1 \circ q_2$

    else $q_1 \circ \langle z.val \rangle \circ q_2$.

Conditional Implementation

- Weaker condition
  - $G \Rightarrow (M^l \Rightarrow M)$
- Additional assumption G:
  - Law of nature, e.g. monitonicity of time in real-time specifications: $(now \in \mathbf{R}) \wedge \Box[now' \in (now, \infty)]_{now}$
  - Interface refinement i.e. relation between low-level tuple l of variables and its high-level representation h, e.g. sequence of bits interpreted as ASCII characters.
  - Assumption about modeling of reality, e.g. interleaving assumption.
- Only needed for open systems.
  - In complete system, G is part of system specifications.

Safety and Closure

- *Finite behavior*
  - Finite sequence of states.
  - Formula F, finite behavior ρ.
  - ρ satisfies F iff ρ can be extended to infinite behavior that satisfies F.
  - $\langle\rangle$ satisfies every behavior.
- *Safety property*
  - Formula that is satisfied by infinite behavior σ iff it is satisfied by every prefix f σ.
  - $Init \land \Box[\mathsf{N}]_v$ is a safety property.
- *Closure*
  - $\mathsf{C}(F)$ = *closure* of F.
  - σ satisfies $\mathsf{C}(F)$ iff every prefix of σ satisfies F.
  - Strongest safety property such that $\models F \Rightarrow \mathsf{C}(F)$

Machine Closure

- (P, L) is *machine-closed*
  - $\mathsf{C}(P \wedge L) \equiv P$.
  - L constrains only infinite behaviors.
- Fairness properties used to write machine-closed specfications:
  - Action A is *subaction* of safety property P iff for every finite behavior $\rho = \langle r_0, \ldots, r_n \rangle$ the following holds:
  - If ρ satisfies P and A is enabled in r_n, then there exists some r_n such that $\langle r_0, \ldots, r_{n+1} \rangle$ satsfies P and $\langle r_n, r_{n+1} \rangle$ is an A step.
  - If A $\Rightarrow$ N, then A is subaction of $Init \wedge \Box[\mathsf{N}]_v$.
- Proposition 1:
  - If P is safety property and L is conjunction of formulas $\mathsf{WF}_w(\mathsf{A})$ and/or $\mathsf{SF}_w(\mathsf{A})$ such that A $\wedge$ $(w' \neq w)$ is a subaction of P, then (P, L) is machine-closed.
  - $\mathsf{C}(Init \wedge \Box[\mathsf{N}]_v \wedge L) = Init \wedge \Box[\mathsf{N}]_v$.

Closure and Hiding

- Special hypotheses:
  - $\mathsf{C}(M_1) \wedge \ldots \wedge \mathsf{C}(M_n) \Rightarrow \mathsf{C}(M_n)$.
  - For proof, first compute $\mathsf{C}(M_i)$.
  - Can be done by Proposition 1 for formulas with no internal variables.
  - For formulas with internal variables, use Proposition 2 to allow application of Proposition 1.
- Proposition 2:
  - Let $x, x_1, \ldots, x_n$ be tuples of variables such that no variable in x_i occurs in M or in any M_j with $i \neq j$.
  - If $\models \wedge_{i=1}^{n}$ $\mathsf{C}(\mathsf{M}_i) \Rightarrow \exists x\colon\ \mathsf{C}(M)$,
    then $\models \wedge_{i=1}^{n}$ $\mathsf{C}(\exists x_i\colon\ \mathsf{M}_i) \Rightarrow \mathsf{C}(\exists x\colon\ M)$.

Additional Temporal Operators

- E_{+v}
  - If E ever becomes false, then state function v stops changing.
  - Behavior σ satisfies E_{+v} if either σ satsfies E or E holds for the first n states of $\sigma$and v never changes form $(n+1)$-st state on.
- Proposition 3:
  - $\models (\exists x\colon Init \land \Box[\mathsf{N}]_w)_{+v} =$
    $\exists x, s\colon \widehat{Init} \land \Box[\widehat{\mathsf{N}}]_{\langle w,v,s\rangle}$
  - x tuple of variables not occuring in v.
  - s variable not occuring in $Init$, N, w, v, x.
  - $\widehat{Init} \equiv (Init \land (s=0)) \lor (\neg Init \land (s=1))$
  - $\widehat{\mathsf{N}} \equiv ((s=0) \land (((s'=0) \land (\mathsf{N} \lor (w'=w)))$
    $\lor ((s'=1) \land \neg(\mathsf{N} \lor (w'=w)))))$
    $\lor ((s=1) \land (s'=1) \land (v'=v))$
- Reasoning about E_{+v}
  - Calculate $\mathsf{C}(E)_{+v} = \mathsf{C}(E_{+v})$ applying Prop. 3.

Additional Temporal Operators

- $E \overset{+}{-\!\triangleright} M$
  - M holds at least as long as E does.
  - Behavior σ satisfies $E \overset{+}{-\!\triangleright} M$ iff it satisfies $E \Rightarrow M$ and if ρ satisfies E then ρ satisfies M (for every finite prefix ρ of σ).
  - $E \overset{+}{-\!\triangleright} M \equiv (\mathsf{C}(E) \overset{+}{-\!\triangleright} \mathsf{C}(M) \wedge (E \Rightarrow M)$.
- $E \overset{+}{-\!\triangleright} M$
  - M is guaranteed under assumption E.
  - M holds at least one step longer as E does.
  - σ satisfies $E \overset{+}{-\!\triangleright} M$ iff it satisfies $E \Rightarrow M$ and if E holds for the first n states of σ, then M holds for the first $n+1$ states of σ.
  - $E \overset{+}{-\!\triangleright} M \equiv (\mathsf{C}(E) \overset{+}{-\!\triangleright} \mathsf{C}(M) \wedge (E \Rightarrow M)$.

Additional Temporal Operators

- $E \perp M$
  - E is *orthogonal* to M.
  - No step will make both E and M false.
  - Behavior σ satsfies $E \perp M$ iff there is no $n \geq 0$ such that both E and M are true for the first n states of σ and false for the first $n+1$ states of σ.
  - $E \perp M \equiv (\mathsf{C}(E) \perp M)$
- Relationships:
  - $(E \perp M) \Rightarrow ((E \overset{+}{\rightarrow\!\!\triangleright} M) \equiv (E \overset{+}{\rightarrow\!\!\triangleright} M))$
  - $(E \overset{+}{\rightarrow\!\!\triangleright} M) \equiv (E \overset{+}{\rightarrow\!\!\triangleright} M) \wedge (E \perp M)$

Further Propositions

- E, M safety properties:
  - $E \stackrel{+}{\rightharpoonup} M \equiv E_{+v} \stackrel{+}{\rightharpoonup} M$ (v containing all free varaibles of M)
  - $E \perp M \equiv (E \land M) \stackrel{+}{\rightharpoonup} (E \lor M)$
- Proposition 4
  - E, M, R safety properties.
  - v containing all free variables of M.
  - $(\models E \land R \Rightarrow M)$ and $(\models R \Rightarrow E \perp M)$ imply $(\models E_{+v} \land R \Rightarrow M)$.
  - Use orthogonality to remove $+$.
- Proposition 5
  - Assume $\models \mathsf{C}(E) = Init_E \land \Box[\mathsf{N}_E]_{\langle x,e \rangle}$
  - Assume $\models \mathsf{C}(M) = Init_M \land \Box[\mathsf{N}_M]_{\langle y,m \rangle}$
  - Then $\models (\exists x\colon Init_E \lor \exists y\colon Init_M)$
    $\land\ Disjoint(e,m)$
    $\Rightarrow \mathsf{C}(\exists x\colon E) \perp \mathsf{C}(\exists y\colon M)$