Richer Interface Automata
with Optimistic and Pessimistic Compatibility

Gerald Lüttgen
Software Technologies Research Group
University of Bamberg, Germany

(Joint work with Walter Vogler, University of Augsburg, Germany)
Interface Theories

• Traditional interfaces *(from 1960s)*
  – Deal with *types of data and operations*
  – Are implemented in most compilers

• Sequential, object-oriented interfaces *(from mid 1980s)*
  – *Behavioural types*, as supported by automated verification tools
  – E.g., pre-/post-conditions and invariants of classes and methods
  – Prominent example: *Meyer‘s contracts in Eiffel*

• Concurrent, behavioural interfaces *(from early 2000s)*
  – *Take concurrency into account*
  – Describe the interaction between system components
  – Prominent prototype: *de Alfaro/Henzinger‘s Interface Automata (IA)*
Semantic Framework

• **Labelled transition systems** representing component behaviour
  
  – *States* $p$ connected via *transitions* $p' = a \rightarrow p''$
  
  – Transitions labelled with *input* ($i?$), *output* ($o!$) or *internal* ($\tau$) *actions*, taken from some alphabets

• **Desiderata**
  
  – Parallel composition
    
    • $p | q$ with a sensible notion of *component compatibility*

  – Compositional refinement preorder
    
    • $p \leq q$ implies $p | r \leq q | r$ (monotonicity of parallel composition)

  – Conjunction
    
    • Supporting perspective-based specification
    
    • $r \leq p \& q$ if and only if $r \leq p$ and $r \leq q$ (greatest lower bound)

  – Alphabet extension
    
    • Permitting the addition of new `features´ when refining interfaces
Parallel Composition on IA

• Basic idea adopted from process algebra (cf. Milner’s CCS)
  – *Synchronize on shared actions* (a!, a?), *resulting in the internal action* $\tau$
  – *Interleave on non-shared actions*

• Example

```
\begin{align*}
p & \xrightarrow{a!} b? \xrightarrow{c!} d! \xrightarrow{\tau} E & q & \xrightarrow{a?} \xrightarrow{d?} e! \\
p \parallel q & \xrightarrow{\tau} b? \xrightarrow{c!} E & &
\end{align*}
```

• Optimistic notion of compatibility

  – *Inputs can be delayed, outputs are immediate*
  – If output of $p$ is not expected by $q$, then $p \parallel q$ is an *error state*
  – Remove states that can reach an error locally (i.e., via outputs or $\tau$s)
IA-Refinement

• An alternating simulation: the largest relation ≤ on states s. t. for all p ≤ q
  – p – a! → p’ implies ∃ q’. q (– τ –>)* – a! → q’ and p’ ≤ q’
  (and similarly for τ-actions)

• Intuition
  – Specified inputs (outputs) must be implemented (are not required)
  – Unspecified inputs (outputs) are always allowed (are forbidden)

• Compositionality result for input-deterministic IA
  – p ≤ q implies p|r ≤ q|r for any input-deterministic r
  – Direct matching of inputs (i.e., without leading τs) is important
Conjunction on IA

- Example with interfaces that have the same alphabets

- Any set of interfaces is implementable (no inconsistencies)

- But BlackHole also means that IA cannot enforce outputs !!!

\[ p \land q \leq p \] (for any interface automaton \( p \))
I/O Modal Transition Systems (IOMTS)

- Adopt idea of modal transition systems (MTS) [Larsen, 1990s]
  - Add a must-/may-modality to each action-labelled transition
    - Must- (may-)transitions must (may) be implemented
    - Unspecified transitions are prohibited in any implementation/refinement
  - Syntactic consistency, i.e., every must-transition is also a may-transition

- IOMTS devised by Larsen, Nyman and Wasowski [ESOP`07]
  - MTS with input and output, as well as classic modal-refinement
  - Parallel composition and compatibility as in IA
  - **But:** modal-refinement is not a precongruence for parallel composition

- Modal-refinement is again an alternating simulation
  - Must-transitions in (IO)MTS are treated as input-transitions in IA
  - May-transitions in (IO)MTS are treated as output-transitions in IA
  - **But:** τ-transitions on the specification side are ignored in IOMTS
Conjunction on MTS (*no IO*)

- **Example**

  (cf. Benes, Cerna and Kretinsky [ATVA‘11])

  After `upper` action $c$, something impossible (action $d$) must be done.
  Inconsistencies are pruned similar as in parallel composition for IA.

- Conjunction in the presence of $\tau$-transitions is treated in [L., V., 2012]
Modal Interface Automata (MIA)

- Inherits from MTS
  - Disjunctive must-transitions incl. \(\tau\)-must-transitions, unlike in IOMTS

- Inherits from IA
  - Parallel composition, including the notion of compatibility
  - Input-determinism, but relaxed due to disjunctive input-must-transitions

- Merges ideas from MTS and IA
  - Refinement as in MTS, but inputs are always allowed as in IA
    - All specified inputs are must-inputs
    - Parallel composition is monotonic, unlike in IOMTS
    - Must-\(\tau\)s on specification side are considered, unlike in IOMTS
  - Conjunction, where inputs (outputs) are treated as in IA (MTS)

- Adds alphabet extension
  - As advocated by Raclet, Caillaud et al. [Fund. Inform., 2011]
MIA-Refinement

• Refinement preorder
  – **Considers** τ-must-transitions on the specification side
  – *Permits alphabet extension on the implementation side*
  – *Another contribution*: definition of weak disjunctive must-transitions

• MIA-refinement ≤
  
  Given MIAs P, Q s.t. I_P ⊇ I_Q, O_P ⊇ O_Q and p≤q

  1. q – i → Q’ implies ∃ P’. p – i → P’ and ∀ p’∈P’ ∃ q’∈Q’. p’≤q’
  2. q – o → Q’ implies ∃ P’. p = o ⇒ P’ and ∀ p’∈P’ ∃ q’∈Q’. p’≤q’
  3. q – τ → Q’ implies ∃ P’. p = ε ⇒ P’ and ∀ p’∈P’ ∃ q’∈Q’. p’≤q’

  4. p – i → P’ and i ∉ I_Q implies ∀ p’∈P’. p’≤q
  5. p – o → p’ and o ∈ O_Q implies ∃ q’. q :: o ::> q’ and p’≤q’
  6. p – o → p’ and o ∉ O_Q implies ∃ q’. q :: ε ::> q’ and p’≤q’
  7. p – τ → p’ implies ∃ q’. q :: ε ::> q’ and p’≤q’
Conjunction on MIA

- Conjunction for interfaces with the same alphabet
  - Treatment of inputs as for IA and of outputs and $\tau$s as for MTS

- But problem with conjunction for dissimilar alphabets

  $p \{o!\} \quad o! \quad o! \quad q \{i?\} \quad o!$

  $u \{i?, o!\} \quad i? \quad o! \quad v \{i?, o!\} \quad o! \quad w \{i?, o!\} \quad o!$

- Extending alphabets morally means to add `neutral´ may-loops
  - Here, $p \land q$ should intuitively be implementable by $u$ and $v$ but not $w$
  - No MIA has $u$ and $v$ but not $w$ as implementations, since it cannot have an initial $i?$-must-transition because of $v$ !!!
The Pessimistic Approach

- Restrictive, pessimistic notion of compatibility
  - Advocated by Bauer, Hennicker et al. in their MIO theory [TACAS’10]
  - If any error state in a parallel composition can be reached via any actions (including input actions), then the composition is not defined
  - This approach distinguishes between $\bullet \xrightarrow{i?} \bullet_E$ and $\bullet \xrightarrow{i?}$

- Trade-off
  - Classic view of may-inputs (i.e., input allowed or prohibited)
  - Much fewer parallel compositions are permitted

- Investigating MIA for pessimistic compatibility
  - Here, one can do away with input-determinism and with the requirement that inputs are must-inputs
  - ‘Standard´ modal-refinement, with compositional parallel operator
Conjunction with Alphabet Extension

- **Conjuncts with dissimilar alphabets**
  
  
  - Conjunction with dissimilar alphabets
    
    \[ p \quad a? \xrightarrow{} b! \xrightarrow{} \]
    
    \[ q \quad \]
    
    
  - **Alphabet extension** \([p]_{c!}\) of conjunct \(p\)
    
    \[ [p]_{c!} \quad a? \xrightarrow{} b! \xrightarrow{} \]
    
    \[ [q]_{\{} \quad a? \xrightarrow{} c! \xrightarrow{} \]
    
    
  - **Conjunctive composition**
    
    \[ p \land q = [p]_{c!} \land [q]_{\{}} \]
    
    
    © 2013 G. Lüttgen, W. Vogler

AVoCS'13 – Richer Interface Automata – 13
Conclusion: Richer Interface Automata

• **MIA is a compositional interface theory that allows one to**
  - Enforce outputs
  - Express disjunctive must-transitions
  - Specify non-deterministic behaviour (unlike Raclet et al. [Fund. Inform., 2011])
  - Abstract from internal computation
  - Compose interfaces in parallel, conjunctively and disjunctively
  - Interpret compatibility optimistically or pessimistically
  - Extend alphabets while refining

• **Some questions left to future work**
  - Are there ‘clean’ interface theories in-between the optimistic and pessimistic approaches?
  - *Is there a possibility to allow may-inputs while maintaining a true open-systems view?*
Selected Literature


