Contract Theories for Embedded Systems: users' requirements, failure or success to meet them, and a new proposal

A team work presented by Albert Benveniste, INRIA

WFCD, ESWEEK 2009, Grenoble

Contract Theories for Embedded Systems:
1: users' requirements, failure or success to meet them
2: a new proposal

Part 1 of the talk: Eric Badouel†, Albert Benveniste†, Benoît Caillaud†, Tom Henzinger§, Axel Legay†, and Roberto Passerone‡

Part 2 of the talk: Benoît Caillaud and Jean-Baptiste Raclet†

†: INRIA
§: EPFL
‡: University of Trento

Users’ Requirements

Positioning contract-based design in embedded systems design flow

- Detailed system design
  - Enabling separate development of components
  - Roles and duties of component vs. environment made explicit
  - Handling detailed design models, functional & extra-functional
  - Focus on scope, power, and computational cost of analysis

- Early requirements capture
  - Contracts as legal bindings for OEM-supplier chains; explicit assumptions and guarantees
  - Enabling separate development of *components* and *viewpoints*; facilitating integration
  - Should accommodate existing design flows & system architectures

Positioning contract-based design in embedded systems design flow: our focus

- Detailed system design
  - Enabling separate development of components
  - Roles and duties of component vs. environment made explicit
  - Handling detailed design models, functional & extra-functional
  - Focus on scope, power, and computational cost of analysis
- Early requirements capture
  - Contracts as legal bindings for OEM-supplier chains; explicit assumptions and guarantees
  - Enabling separate development of *components* and *viewpoints*; facilitating integration
  - Should accommodate existing design flows & system architectures

Positioning contract-based design in embedded systems design flow: refining requirements

- Explicit roles
  - Component/Environment
  - Assumptions/Guarantees

- Early requirements capture
  - Contracts as legal bindings for OEM-supplier chains; explicit assumptions and guarantees
  - Enabling separate development of *components* and *viewpoints*; facilitating integration
  - Should accommodate existing design flows & system architectures

Positioning contract-based design in embedded systems design flow: refining requirements

- Explicit roles
  - Component/Environment
  - Assumptions/Guarantees
- Conjunctive requirements
  - Multiple viewpoints
  - Doors/Excel Req capture
- Early requirements capture
  - Contracts as legal bindings for OEM-supplier chains; explicit assumptions and guarantees
  - Enabling separate development of *components* and *viewpoints*; facilitating integration
  - Should accommodate existing design flows & system architectures

Positioning contract-based design in embedded systems design flow: refining requirements

- Explicit roles
  - Component/Environment
  - Assumptions/Guarantees
- Conjunctive requirements
  - Multiple viewpoints
  - Doors/Excel Req capture
- Allow for flexible design flow
  - Component first vs. viewpoint first
  - System/service Architecture ≠ Execution Infrastructure
- Early requirements capture
  - Contracts as legal bindings for OEM-supplier chains; explicit assumptions and guarantees
  - Enabling separate development of *components* and *viewpoints*; facilitating integration
  - Should accommodate existing design flows & system architectures

Positioning contract-based design in embedded systems design flow: refining requirements

- Explicit roles
  - Component/Environment
  - Assumptions/Guarantees
- Conjunctive requirements
  - Multiple viewpoints
  - Doors/Excel Req capture
- Allow for flexible design flow
  - Component first vs. viewpoint first
  - System/service Architecture ≠ Execution Infrastructure
- Locality

- Early requirements capture
  - Contracts as legal bindings for OEM-supplier chains; explicit assumptions and guarantees
  - Enabling separate development of *components* and *viewpoints*; facilitating integration
  - Should accommodate existing design flows & system architectures

Some frameworks for contract-based design

Some frameworks for contract-based design: they all offer provision for

- Attaching contracts & implementations to components
  - Component / Environment
  - (A,G) = (Assume, Guarantee)
- Composing components ⊗
- Additional services
  - Well-formedness
  - Deadlock avoidance

- Satisfaction

  $M \models C$ if when put under any E meeting assumptions specified in C, implementation M satisfies guarantees entailed by C
- Consistency

  C is consistent if it admits a non-empty implementation
- Compatibility

  C is compatible if it admits a non-empty environment
- Refinement

  $C' \leq C$ if C' has less implementations and more environments that C

Some frameworks for contract-based design

- C=(A,G), M where A,G,M are properties [SPEEDS]
  - A,G explicit, ¬X needed
  - Deadlock etc not considered
- Interface Automata & variants [de Alfaro-Henzinger]
  - A,G implicit
  - Illegal states, game approach
- Modal Interfaces & variants [Larsen]
  - A,G “very” implicit, through modalities for transitions: may/must
  - Surprisingly simple and elegant

- Satisfaction

  M ⊨ C if when put under any E meeting assumptions specified in C, implementation M satisfies guarantees entailed by C

- Consistency

  C is consistent if it admits a non-empty implementation

- Compatibility

  C is compatible if it admits a non-empty environment

- Refinement

  C’≤C if C’ has less implementations and more environments that C

Some frameworks for contract-based design

- C=(A,G), M where A,G,M are properties [SPEEDS]
  - A,G explicit, ¬X needed
  - Deadlock etc not considered
- Interface Automata & variants [de Alfaro-Henzinger]
  - A,G implicit
  - Illegal states, game approach
- Modal Interfaces & variants [Larsen]
  - A,G “very” implicit, through modalities for transitions: may/must
  - Surprisingly simple and elegant

Embeddings exist

Some frameworks for contract-based design

- Interface Automata & variants [de Alfaro-Henzinger]
  - A,G implicit
  - Illegal states, game approach
- Modal Interfaces & variants [Larsen]
  - A,G “very” implicit, through modalities for transitions: may/must
  - Surprisingly simple and elegant

- Reflect the game nature of Interface Automata as follows:
  - Everything the Env is allowed to submit *must* be accepted by the component
  - The Comp *may* output what is allowed by the interface
  - Everything the Env is disallowed to submit leads to a trap (exception state, where anything can happen afterwards)

Some frameworks for contract-based design

- C=(A,G), M   where A,G,M are properties [SPEEDS]
  - A,G explicit, ¬X needed
  - Deadlock etc not considered
- Modal Interfaces & variants [Larsen]
  - A,G “very” implicit, through modalities for transitions: may/ must
  - Surprisingly simple and elegant

- C is characterized by its set of implementations: $\{ M \mid E \subseteq A \Rightarrow M{\times}E \subseteq G \}$
- Regard A and G as Modal Interfaces (with all transitions being *must*); finding C amounts to solving for X the equation $X \otimes A = G$
- Solution: X=G/A the residuation (or quotient) of G by A
- Contracts as quotients G/A

Failure or success to meet users' requirements

Two issues that proved surprisingly critical

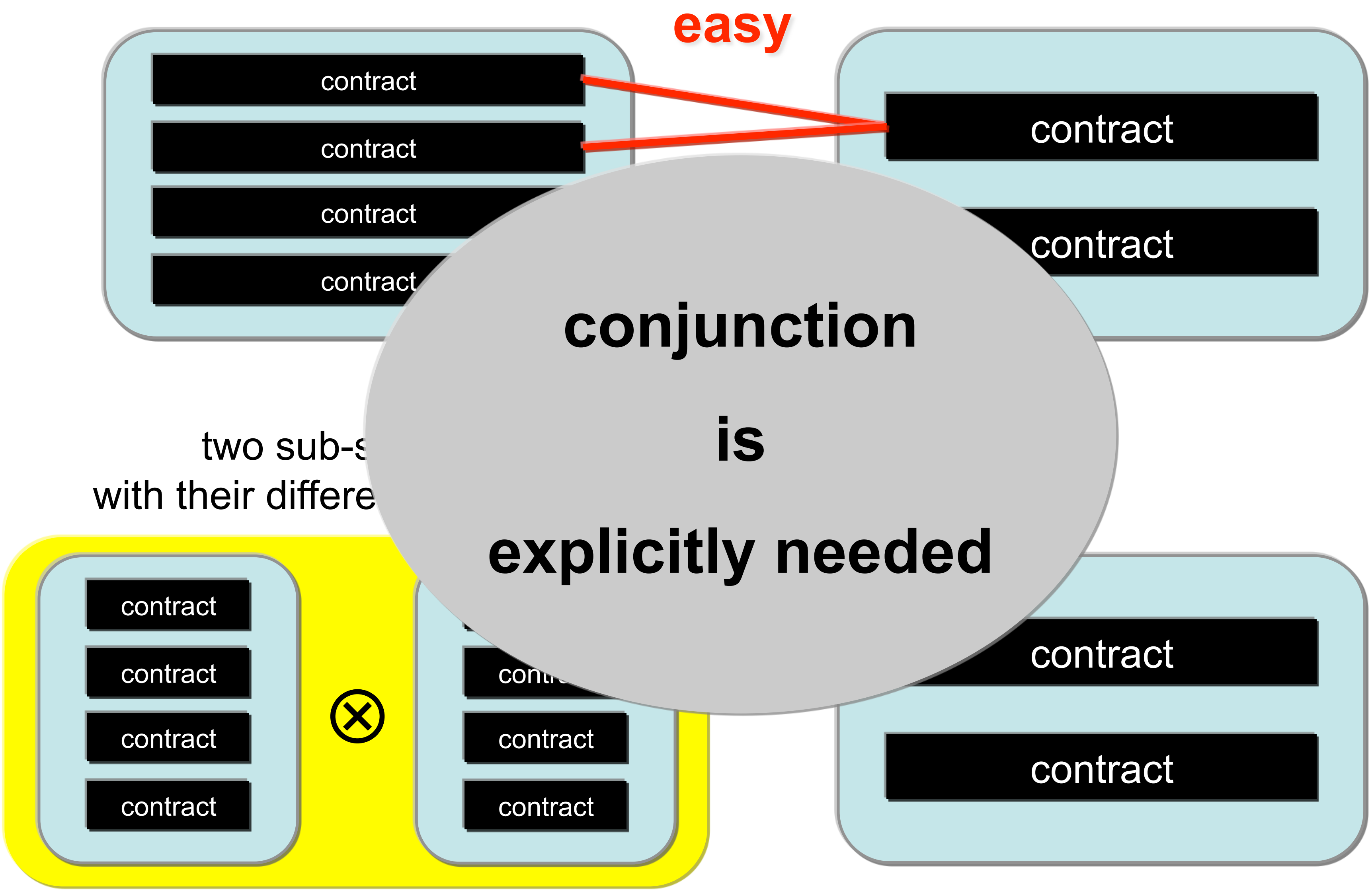
Two issues that proved surprisingly critical

- **Conjunctive Contracts**
  - Followed from users' requirements
  - Allow for flexible design flow
- Questions:
  - Is conjunction explicitly needed or does there exist a turn around?
  - How do the different frameworks support conjunction?

- **Locality of alphabets of ports and actions**
  - Seems like a little technical detail, but still needed
- Question:
  - How do the different frameworks support alphabet equalization?

The issue of Conjunctive Contracts

illustration on the case of refinement



The issue of Conjunctive Contracts

- C=(A,G), M where A,G,M are properties [SPEEDS]
  - A,G explicit, ¬X needed
  - Deadlock etc not considered

- $M \models (A,G)$ iff $M \cap A \subseteq G \Leftrightarrow M \subseteq G \cup \neg A$
- $(A',G') \leq (A,G)$ iff $A' \subseteq A$ and $G' \subseteq G$
  - Refinement is sound but not complete!
- $(A',G') \wedge (A,G) = (A' \cup A , G' \cap G)$
- $(A',G') \otimes (A,G) = ((A' \cap A) \cup \neg (G' \cap G) , (G' \cap G))$
- Seems OK

The issue of Conjunctive Contracts

- Interface Automata & variants [de Alfaro-Henzinger]
  - A,G implicit
  - Illegal states, game approach

- Refinement is by alternating simulation: C'≤C iff, from respective initial state
  - Env' can do whatever Env can
  - Comp can do whatever Comp' can

  and the two moves lead to states where the same repeats

- Conjunction ⇔ Shared Refinemt.
  - Very subtle and solved only for a special class of synchronous transition systems [Emsoft09]

- Problematic.
  Alternative: ATL logic?

The issue of Conjunctive Contracts

- Modal Interfaces & variants [Larsen]
  - A,G “very” implicit
  - Surprisingly simple and elegant

- C is an automaton in which transitions are labeled *may* or *must*
- Actions are labeled either ? =Env or !=Comp
- M ⊨ C iff M offers all *must* transitions and some *may* transitions
- Refining: turning some *may* into *must* and removing other *may*’s
- Conjunction: take product structure and intersection of *may* and union of *must*

The issue of alphabet equalization

- Usually not considered a problem: two automata with different alphabets must synchronize on their shared actions and otherwise interleave
- Amounts to equalization by inverse projections: add in each state self-loops with missing symbols
- Take Assume/Guarantee contracts as an example

Apply this to $(A_1,G_1) \wedge (A_2,G_2)$?

- Suppose A_1 is non-trivial, A_2=true and the two contracts possess disjoint alphabets
- Then equalizing by inverse projections yields
  $(A_1,G_1) \wedge (A_2,G_2) =$
  $(\pi^{-1}(A_1) \cup \pi^{-1}(A_2) , -) =$
  $(\pi^{-1}(A_1) \cup \text{true} , -) = (\text{true},G)$
- Although the two contracts do not interact, the second one kills the assumptions of the first one !!!
- Mathematically consistent but highly non satisfactory

The issue of alphabet equalization

- Reason for this artifact is that equalization by inverse projection is not neutral for conjunction (it is neutral for composition)

- **Problem: how can we have two different alphabet extensions that can respectively be**
  - **Neutral for $\otimes$**
  - **Neutral for $\wedge$**

- No solution found so far in the framework of (A,G) contracts

- Suppose A_1 is non-trivial, A_2=true and the two contracts possess disjoint alphabets

- Then equalizing by inverse projections yields
  $(A_1,G_1) \wedge (A_2,G_2) =$
  $(\pi^{-1}(A_1) \cup \pi^{-1}(A_2), -) =$
  $(\pi^{-1}(A_1) \cup \text{true}, -) = (\text{true}, G)$

- Although the two contracts do not interact, the second one kills the assumptions of the first one !!!

- Mathematically consistent but highly non satisfactory

The issue of alphabet equalization

- Reason for this artifact is that equalization by inverse projection is not neutral for conjunction (it is neutral for composition)
- **Problem: how can we have two different alphabet extensions that can respectively be**
  - **Neutral for $\otimes$**
  - **Neutral for $\wedge$**
- No solution found so far in the framework of (A,G) contracts

- Modal Interfaces offer enough flexibility for having a positive answer to this problem
- Strong extension:
  - add in each state *must* self-loops with missing symbols
- Weak extension:
  - add in each state *may* self-loops with missing symbols
- It turns out that strong extension is neutral for $\otimes$ and weak extension is neutral for $\wedge$

The issue of alphabet equalization

- Reason for this artifact is that equalization by inverse projection is not neutral for conjunction
(it is neutral for composition)

- **Problem: how can we have two different alphabet extensions that can respectively be**
  - **Neutral for $\otimes$**
  - **Neutral for $\wedge$**

- No solution found so far in the framework of (A,G) contracts

- We do not know whether there is a solution to this problem in the framework of Interface Automata & variants

Summary of situation

Summary of situation

- Embeddings exist
  - (A,G) contracts → Modal Interfaces ← Interface Automata
- Modal Interfaces address all difficulties in an elegant way
  - They are the best candidate for future developments

A new proposal: Convex Acceptance Interfaces

InterSMV: a tool under development by Benoit Caillaud at INRIA for handling Convex Acceptance Interfaces

Objectives of InterSMV

- Supporting all basic operations
  - Refinement
  - Conjunction
  - Parallel composition
  - Quotient (Residuation)
  - Weak & Strong alphabet extensions
- Supporting fundamental relations:
  - Implementation
  - Consistency
  - Compatibility
- A front-end for NuSMV
- Handling Modal Interfaces
- Supporting both interleaving and synchronous semantics
  - Interleaving: theory well developed
  - Synchronous: new, non-trivial adaptation

The unexpected difficulty

- Modal Interfaces having Synchronous Symbolic Transition Systems as their implementations, i.e.:
- M = (D,Σ,T), where
  - D is a universal domain of values (for simplification), possibly equipped with a distinguished element to encode absence
  - Σ is a finite alphabet of variables, S=D^Σ is the set of *states*
  - $T \subseteq D^\Sigma \times D^\Sigma$ is the symbolic *transition relation*, relating previous and current variables
- This model is not closed under weak alphabet extension: see next counter-example

Modal Synchronous Transition Systems

- Synchronous Implementation: M = (D,V,T), where
  - D is a universal domain of values (for simplification), possibly equipped with a distinguished element to encode absence
  - V is a finite alphabet of variables, $\Sigma = D^V$ is the set of *states*
  - $T \subseteq \Sigma \times \Sigma$ is the symbolic *transition relation*, relating previous and current variables;
- Modal STS: C = (D,V,*may*,*must*), where
  - *may*, *must* $\subseteq \Sigma \times \Sigma$; consistency holds if *must* $\subseteq$ *may*
  - Implementation: $M \models C$ if $\mathit{must} \subseteq T \subseteq \mathit{may}$
  - In particular, the set of implementations is stable under intersection: if $M, M' \models C$ then $\mathit{must} \subseteq (T \cap T') \subseteq \mathit{may}$

The unexpected difficulty: counter-example

- Signature : V = { x: boolean }
- Modal specification : C = « always (must x) $\wedge$ (may x)»
- Possible implementations satisfying the specification: in every state x must be enabled and $\neg$x must be disabled.
- How to extend C to signature W = { x,y: boolean }, so that we are neutral w.r.t. any specification B over W , taken conjunctively?
- We should allow { x.y } or { x. $\neg$y } or {x.y, x. $\neg$y}, and nothing else.
- Problem : this set is not closed under intersection since { x.y } $\cap$ { x. $\neg$y } = $\varnothing$, which is not part of the above set. .
  Thus the above set is not expressible as a modal specification.

What is the problem?

- Modalities are not flexible enough at specifying the allowed transition relations

What is the problem? What is the solution?

- Modalities are not flexible enough at specifying the allowed transition relations

- Relax modalities by considering instead Acceptance Relations

- Acceptance relation = enumeration of the allowed transition relations, from each given source state

- A comprehensive theory of Acceptance Interfaces has been developed by J-B Raclet in his thesis [Raclet PhD 2007]

What is the problem? What is the solution?

- $A \subseteq \Sigma \times 2^{\Sigma}$
  (modal: $must \subseteq may \subseteq \Sigma \times \Sigma$)
- $C \subseteq C'$ iff $A \subseteq A'$
  $C \wedge C' : A \wedge A'$
  $A \otimes A' = \{ X \cap X' \mid X \in A, X' \in A' \}$
  $A/A' = \{ X \mid \forall X' \in A' : X \cap X' \in A \}$
- Strong and weak extensions:
  $\Sigma' = \Sigma \cup \{a\}$
  $A_{\Uparrow\Sigma'} = \{ X \cup \{a\} \mid X \in A \}$
  $A_{\uparrow\Sigma'} = A \cup A_{\Uparrow\Sigma'}$
- Very elegant, but very costly

- Relax modalities by considering instead Acceptance Relations
- Acceptance relation = enumeration of the allowed transition relations, from each given source state
- A comprehensive theory of Acceptance Interfaces has been developed by J-B Raclet in his thesis [Raclet PhD 2007]

What is the problem? What is the solution?

- Are we done? Not quite so

L Due to extensional enumeration, acceptance relations are computationally intractable

J Idea: search for a framework that sits between Modalities and Acceptance Relations:
    - Convex Acceptance Relations
    - They are characterized via their extremal elements
    - Stable under all operations

- Relax modalities by considering instead Acceptance Relations

- Acceptance relation = enumeration of the allowed transition relations, from each given source state

- A comprehensive theory of Acceptance Interfaces has been developed by J-B Raclet in his thesis [Raclet PhD 2007]

What is the problem? What is the solution?

- Are we done? Not quite so

L Due to extensional enumeration, acceptance relations are computationally intractable

J Idea: search for a framework that sits between Modalities and Acceptance Relations:

  - Convex Acceptance Relations
  - They are characterized via their extremal elements
  - Stable under all operations

- The coding of Interval Acceptance Relations:

  $\Sigma = D^V \ni s \;\rightarrow\; \{ tt, ff, \perp, \top \}$

  $[a^-(s),a^+(s)] = tt$ if $s \in a^- \wedge s \in a^+$

  $[a^-(s),a^+(s)] = ff$ if $s \notin a^- \wedge s \notin a^+$

  $[a^-(s),a^+(s)] = \perp$ if $s \in a^- \wedge s \notin a^+$

  $[a^-(s),a^+(s)] = \top$ if $s \notin a^- \wedge s \in a^+$

- With this coding, handling Convex Acceptance Interfaces can be done using NuSMV: tool InterSMV

Some concluding remarks

- Modal interfaces are a very good basis for contract-based design
- Convex Acceptance Interfaces seem a good compromise
- InterSMV is a tool under development at INRIA for handling modal interfaces
- Still, the situation is far from being satisfactory…

COMBEST

Still, the situation is far from being satisfactory…

- They look simple but the devil is in the details
  - They differ for each different framework
  - Authors may even disagree in what they are
  - There are many variations
- While contracts are appealing to the industry, engineers struggle grasping what these relations are
  - an ongoing effort at CESAR SP2

- Satisfaction

  M $\models$ C if when put under any E meeting assumptions specified in C, implementation M satisfies guarantees entailed by C

- Consistency

  C is consistent if it admits a non-empty implementation

- Compatibility

  C is compatible if it admits a non-empty environment

- Refinement

  C’$\leq$C if C’ has less implementations and more environments that C

SEVENTH FRAMEWORK PROGRAMME

Still, the situation is far from being satisfactory…

- Consistency and compatibility look dual
  - and should be dual (Env and Comp should be dual players);
  - unfortunately they are not!
- Conclusion:
  - The theories should be cleaned up to make fundamental relations crystal clear
  - Or alternatively relations should be made flexible as they become clear in most practical cases

- Satisfaction

  M ╞ C if when put under any E meeting assumptions specified in C, implementation M satisfies guarantees entailed by C

- Consistency

  C is consistent if it admits a non-empty implementation

- Compatibility

  C is compatible if it admits a non-empty environment

- Refinement

  C'≤C if C' has less implementations and more environments that C

Still, the situation is far from being satisfactory…

- Consistency and compatibility look dual
  - and should be dual (Env and Comp should be dual players);
  - unfortunately they are not!
- Conclusion:
  - The theories should be cleaned up to make fundamental relations crystal clear
  - Or alternatively relations should be made flexible as they become clear in most practical cases

- Need for cleaner theories
  - Clarify fundamental relations
  - Clean compatibility vs. consistency
  - Be either functional (In→Out) or relational; avoid hybrids
- Need to smoothly embed contracts into requirements engineering

COMBEST

THANK YOU

Well, assuming that
the audience paid proper attention

Did the speaker meet its promises?

SEVENTH FRAMEWORK PROGRAMME