Towards a Theory of Architectural Contracts: **Schemes and Patterns of Assumption/Promise Based System Specification**

Manfred Broy

Technische Universität München
Institut für Informatik
D-85748 Garching, Germany

Contracts

Contracts support following software engineering principles

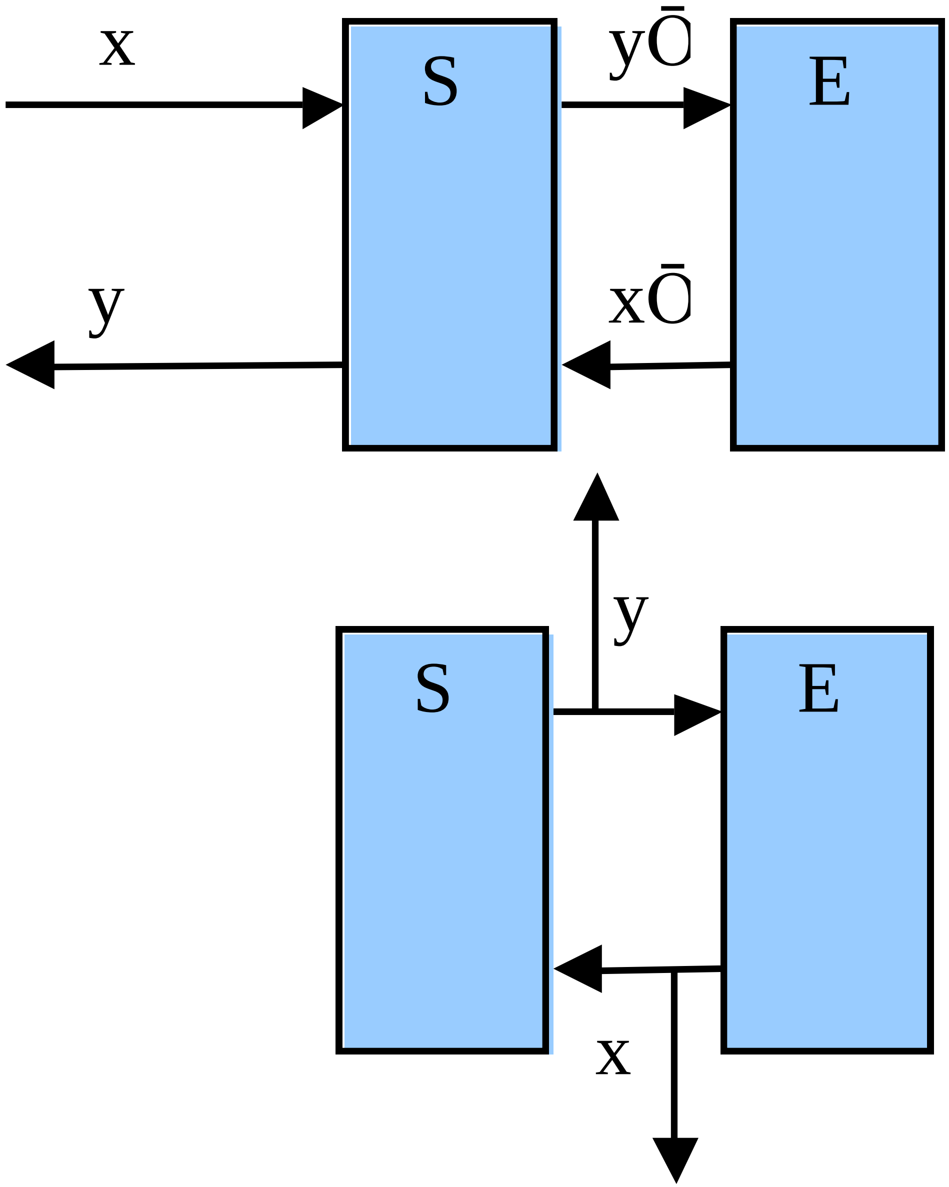
- Modularity
  - ◊ Modular refinement
- Interface abstraction
- State encapsulation
- Information hiding
- Divide and conquer
- Design patterns

Assumption/Promise: Basic Idea

Specification pattern, to formulate a contract:

- If the environment fulfils assumptions, then the system promises (guarantees, is committed to) properties
- This reflects the idea of a contract between
  - the developer of the system
  - the architect that selects the environment for the system
- Contracts include notions of compatibility
  - Under which conditions can a sub-system be replaced by another one with compatible behaviour

Composition



E⊗S

Hiding internal channels

E×S

Visibility of internal channels

A/P-Pattern

- Let System be the set of all systems.
- Composiing system $S \in$ System with environment $E \in Env(S) \subseteq$ System results in

$$E \times S \in \text{System}$$

- Based on composition operator $\times$ we formulate contracts by assumptions and promises:

$$Con(S) \equiv \forall\ E \in Env(S): Asu(E) \Rightarrow Pro(E \times S)$$

  where
  - ◊ $Con(S)$ is a system specification called contract,
  - ◊ $Asu(E)$ is an environment specification called assumption and
  - ◊ $Pro(E \times S)$ is a specification about the system $E \times S$ called a promise.
- The predicates specify properties

$$Con, Asu, Pro: \text{System} \ \rightarrow\ \mathbb{B}$$

Contracts Specifying Functional Properties

Given a syntactic interface (I▸O) an interface assertion

is a Boolean expression p(x, y) where p is a predicate

$$p: \vec{I} \times \vec{O} \rightarrow \mathbb{B} .$$

and $x \in \vec{I}$ and $y \in \vec{O}$ are input and output histories

Semantics of Contracts by Logical Implication

- Interface assertions structured into following pattern:

  assumption: asu(x, y)

  promise: pro(x, y)

  with the meaning: if the environment fulfils the assumption

  asu(x, y)

  then the system fulfils the promise

  pro(x, y)

- We require of environment E the assumption specified by

  $$\text{Asu}(E) \equiv [\forall\, x, y: x \in E(y) \Rightarrow \text{asu}(x, y)]$$

  and of the system S and its environment E the promise is specified by

  $$\text{Pro}(E, S) \equiv [\forall\, x, y: y \in (E \times S)(x) \Rightarrow \text{pro}(x, y)]$$

  The combination of these predicates then specifies a contract

  $$\text{Con}(S) \equiv [\forall\, E: \text{Asu}(E) \Rightarrow \text{Pro}(E, S)]$$

  This defines the meaning of a functional contract.

Deriving Implicative Assertions from Contracts

- We consider the predicates

$$\text{Asu}(E) \equiv [\forall\ x, y: x \in E(y) \Rightarrow \text{asu}(x, y)]$$

$$\text{Pro}(E, S) \equiv [\forall\ x, y: y \in (E \times S)(x) \Rightarrow \text{pro}(x, y)]$$

- The combination of these predicates specifies a contract

$$\text{Con}(S) \equiv [\forall\ E: \text{Asu}(E) \Rightarrow \text{Pro}(E, S)]$$

which unfolds into

$$\text{Con}(S) \equiv$$

$$[\forall\ E: [\forall\ x, y: x \in E(y) \Rightarrow \text{asu}(x, y)] \Rightarrow$$

$$[\forall\ x, y: y \in (E \times S)(x) \Rightarrow \text{pro}(x, y)]]$$

- The restriction of causality and realizability for environment E and S allows us to derive further properties.

Causality and Realizability

- In case assertion $asu(x, y)$ is causal and fully realizable there exists a most general environment E_{gen} such the following property holds :

  $$\forall\ x, y: x \in E_{gen}(y) \Leftrightarrow asu(x, y)$$

- If a most general environment exists, then

  $$Con(S) \equiv [\forall\ x, y: y \in (E_{gen} \times S)(x) \ \Rightarrow pro(x, y)]$$

  This semantic interpretation of the A/P pattern is equivalent to

  $$Con(S) \equiv [\forall\ x, y: y \in S(x) \wedge x \in E_{gen}(y) \Rightarrow pro(x, y)]$$

  which leads by the specification of E_{gen} to the following contract:

  $$Con(S) \equiv \forall\ x, y: asu(x, y) \Rightarrow (y \in S(x) \Rightarrow pro(x, y))$$

  and to interface assertion $con(x, y)$ for contract $Con(S)$

  $$con(x, y) \equiv [asu(x, y) \Rightarrow pro(x, y)]$$

Assumptions have to Speak about Output

- Consider a system with input channel x and output channel y which numbers as messages specified by

  $$asu(x, y) \equiv \forall\ t: \forall\ n \in IN: n\#(x{\downarrow}t) \leq (n\#y{\downarrow}t)+1$$

  $$pro(x, y) \equiv \forall\ n \in IN: n\#x = n\#y$$

- We get the specification in terms of an interface assertion

  $$con(x, y) \equiv [asu(x, y) \Rightarrow pro(x, y)]$$

- The promise is only guaranteed if a next copy of a number n is never sent to the system before the copy previously sent has been forwareded.

Implicative interface assertions

Tab. 1 Cases of Validity of con(x, y), asu(x, y), and pro(x, y)

con(x, y)	asu(x, y)	pro(x, y)	Interpretation
true	true	true	for system S history y is a correct output for valid input history x
false	true	false	for system S history y is not a correct output for valid input history x
true	false	true	for system S and history y input history x is not a valid input
true	false	false	for system S and history y input history x is not a valid input

Architectural Contracts

- Given an A/P specification

  assumption: $asu(x, y)$

  promise: $pro(x, y)$

  one interpretation is that the system S is only used in environments E where assumption $asu(x, y)$ holds.

  Then we get

  $$asu(x, y) \wedge pro(x, y)$$

  This interpretation is called *architectural contract*.

Implicative Assertions

- A derived interpretation is an implicative assertion

  $$\text{con}(x, y) \equiv [\text{asu}(x, y) \Rightarrow \text{pro}(x, y)]$$

  that specifies the properties implied for system S by the A/P specification.
- If system S is only used in environments E with specifying assertion env(x, y) we get by composition for the composite system $E \times S$

  $$\text{env}(x, y) \wedge (\text{asu}(x, y) \Rightarrow \text{pro}(x, y))$$

  which is different to the architectural contract interpretation.

Example: General Implicative Assertions

- Let n be a given natural number.
- Consider a system with input channel x and output channel y, both carrying natural numbers as messages with specification

$$\mathrm{con}(x, y) \equiv [n\#y = 0 \Rightarrow n\#x = 0]$$

- The premise is not a meaningful assumption, since
  - there does not exist an environment that guarantees assertion $n\#y = 0$
  - it does not speak about input x but only about output y.
- Assertion $n\#y = 0$ is not causal in history y, since

$$y{\downarrow}t = y'{\downarrow}t \Rightarrow \forall\ x: (n\#y = 0) \equiv (n\#y' = 0)$$

  which does not hold.
- Assertion $n\#y = 0$ is not a healthy assumption, since it is not realizable by any environment.

Example: Implicative Assertions (ctd)

- Assertion

  $$con'(x, y) \equiv [n\#x > 0 \Rightarrow n\#y > 0]$$

  is equivalent to assertion $con(x, y)$ by contraposition
- Assertion $n\#x > 0$ is causal in history y since the formula

  $$y{\downarrow}t = y'{\downarrow}t \Rightarrow \forall x: (n\#x{\downarrow}t > 0) \equiv (n\#x{\downarrow}t > 0)$$

  holds.
- This assertion may be interpreted as an A/P-format

  assumption: $n\#x > 0$

  promise: $n\#y > 0$

  which is a meaningful (but rather simple) contract.

Healthiness Conditions for Contracts

Useless Contracts

- There are two cases of contracts Con(S) that are not very useful:

  Con(S) = true

  and

  Con(S) = false

- In the first case we speak of a trivial specification in the second case of a paradoxical specification.

Non-satisfiable Specifications

- We call assumption Asu(E) about environment E non-satisfiable if there does not exist some environment E such that Asu(E) holds.

  Then contract Con(S) is trivial.
- Let Asu be specified based on asu as defined above.
- If asu(x, y) is false, then Asu is non-satisfiable.
- Even in cases where asu(x, y) is not identical to false, predicate Asu may be non-satisfiable.

Non-satisfiable Specifications

Theorem:

- If every environment E can be represented by a total Mealy machine, then Asu(E) is satisfiable if and only if asu(x, y) is realizable (for the environment with input y and output x).
- Proof:
  For a specification asu(x, y) there exists a Mealy machine that satisfies asu(x, y) if and only if asu(x, y) is realizable.

Safety and Liveness of Interface Assertions

Technische Universität München
Institut für Informatik
D-85748 Garching, Germany

Safety

- A predicate R is a pure safety property if the following equivalence holds for all histories x and y:

  $$R(x, y) \equiv \forall\ t: R(x{\downarrow}t, y{\downarrow}t)$$

- Since always the following condition holds

  $$(\forall\ t: R(x{\downarrow}t, y{\downarrow}t)) \Leftarrow R(x, y)$$

- R is a safety property iff for all histories x and y):

  $$(\forall\ t: R(x{\downarrow}t, y{\downarrow}t)) \Rightarrow R(x, y)$$

Liveness

- R is a pure liveness property if

$$\forall\ t: R(x{\downarrow}t, y{\downarrow}t)$$

- The only predicate that is both a pure safety and a pure liveness predicate is the predicate true.

Decomposing Assertions into Safety and Liveness

- The safety part R^* of an interface assertion $R(x, y)$ is given by the following equation

  $$R^*(x, y) \equiv \forall\ t: R(x\downarrow t, y\downarrow t)$$

- R is called safety realizable if:

  $$\forall\ x: \exists\ y: R^*(x, y)$$

- For predicate R we get liveness property R^∞ included in property R by

  $$R^\infty(x, y) \equiv (\neg R^*(x, y) \vee R(x, y))$$

- To show that R^∞ is a liveness property we have to prove

  $$\forall\ t: R^\infty(x\downarrow t, y\downarrow t)$$

Assumption/Promises as Safety and Liveness Properties

- We consider the interface assertion con(x, y) with

$$con(x, y) \equiv [asu(x, y) \Rightarrow pro(x, y)]$$

- The liveness conditions in assertion asu(x, y) for input history x may depend on safety properties of y.
- A typical example would be
  - ◊ If y(t) is a query, then there exists a time t' > t such that x(t') is a reply to this query.

Assumption asu and promise pro as safety property

- In this case the A/P-scheme is equivalent to the following assertion:

  $$\text{con}(x, y) \equiv \forall\ t: [\text{asu}(x{\downarrow}t, y{\downarrow}t) \Rightarrow \text{pro}(x{\downarrow}t, y{\downarrow}t)]$$

  This is the consequence of the required causality of con(x, y).

Assumption asu as safety, promise pro as liveness property

- In this case the A/P-specification con(x, y) is equivalent to the following assertion:

$$\text{con}(x, y) \equiv [\forall\ t: \text{asu}(x{\downarrow}t, y{\downarrow}t)] \Rightarrow \text{pro}(x, y)$$

  This is the consequence of the required causality of con(x, y).

Assumption asu as liveness, promise pro as safety property:

- In this case we can strengthen the specification according to realizability on con(x, y)

$$\text{con}(x, y) \equiv \text{pro}(x, y)$$

- Since the violation of assumption asu(x, y) cannot be observed in finite time, but promise pro can only be violated in finite time, a computation strategy has to observe promise pro in any case.

An example

$$\mathrm{asu}(x, y) \equiv (\mathrm{true}\#x = \infty)$$

$$\mathrm{pro}(x, y) \equiv (\mathrm{true}\#y = 0)$$

Assume a realization f that

fulfils $\mathrm{true}\#f(x) > 0$ for some x with $\mathrm{true}\#x = n \in \mathrm{IN}$.

This leads to a contradiction since by

$\mathrm{true}\#f(x) > 0$ there exists some t with

$\mathrm{true}\#f(x){\downarrow}t > 0$ and thus for history x' with

$x{\downarrow}t = x'{\downarrow}t$ and $\mathrm{true}\#x' > \infty$

we get $\mathrm{true}\#f(x') > 0$ which violates the specification

$$\mathrm{true}\#x = \infty \Rightarrow \mathrm{true}\#f(x) = 0$$

Assumption asu and promise pro as liveness properties

- In this case the condition

  $$asu(x, y) \Rightarrow pro(x, y)$$

  can be fulfilled by fulfilling promise $pro(x, y)$ in any case.
- Otherwise, the liveness condition have to fit together.

An example

$$asu(x, y) \equiv (true\#x = \infty)$$

$$pro(x, y) \equiv (true\#y < \infty)$$

Decomposing A/P Specification Into Safety and Liveness

- We decompose assumption asu and promise pro into pure safety properties asu_S, pro_S and pure liveness properties asu_L and pro_L such that

$$con(x, y) \equiv$$

$$[asu_S(x, y) \wedge asu_L(x, y) \Rightarrow pro_S(x, y) \wedge pro_L(x, y)]$$

- For a strongly causal and realizable specification $con(x, y)$ we can derive specific assertions

$$asu_S(x, y) \Rightarrow pro_S(x, y)$$

$$asu_S(x, y) \wedge asu_L(x, y) \Rightarrow pro_L(x, y)$$

Conclusion

- Analysing the assumption/promise pattern additional consequences are derived by
  - ◊ Causality and realizability requirements
  - ◊ Safety and liveness considerations