Contract Modeling

Christian Stefansen and Philipp Kutter

Montages partner meeting Sep. 1, 2007

Does your company systematically meet its contractual obligations?

Montages tm
faster development

Does your company loose money due to missed financial opportunities?

Can you exchange contract information seamlessly between front- and back-office?

Modeling contracts

(and good contract management systems based on these models)

can ensure this!

ContractML

ContractML is a

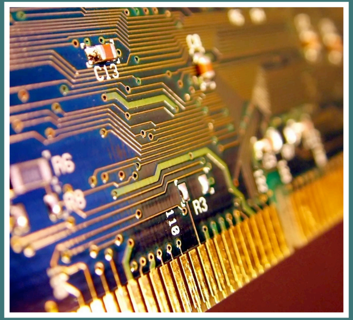
- proven,
- research-based,
- domain-specific language (DSL)

for modeling contracts.

ContractML is work by Jesper Andersen, Ebbe Elsborg,
Jakob Grue Simonsen, Christian Stefansen, and Fritz Henglein

Agenda

The business case



The technology

Case studies

Montages tm
faster development

The business case

What is contract management?

- Write, maintain, monitor, and analyze contracts:
  - Create new types of contracts
  - Manage execution dates for rights and obligations (scheduling)
  - Compute pricing/volatility for standard and custom-made financial instruments.
  - Generic deal-capturing, portfolio management, and trading agents.
  - Analyze, integrate, and monitor risks (operational, credit, market)

Business drivers

Business drivers

- Business cycle is getting shorter: demand for fast implementation of new exotic instruments

Business drivers

- Business cycle is getting shorter: demand for fast implementation of new exotic instruments
- Financial companies compete on continuous and precise valuation of instruments

Business drivers

- Business cycle is getting shorter: demand for fast implementation of new exotic instruments
- Financial companies compete on continuous and precise valuation of instruments
- Cost reduction pressure to integrate systems front-to-back and with partners

Business drivers

- Business cycle is getting shorter: demand for fast implementation of new exotic instruments
- Financial companies compete on continuous and precise valuation of instruments
- Cost reduction pressure to integrate systems front-to-back and with partners
- Autonomous trading agents are becoming important to react immediately on fluctuations

Contract modeling today

Contract modeling today

- Non-existent (paper-based)
  - Manual valuation/risk analysis is error-prone and slow
  - Easy to miss deadlines and opportunities

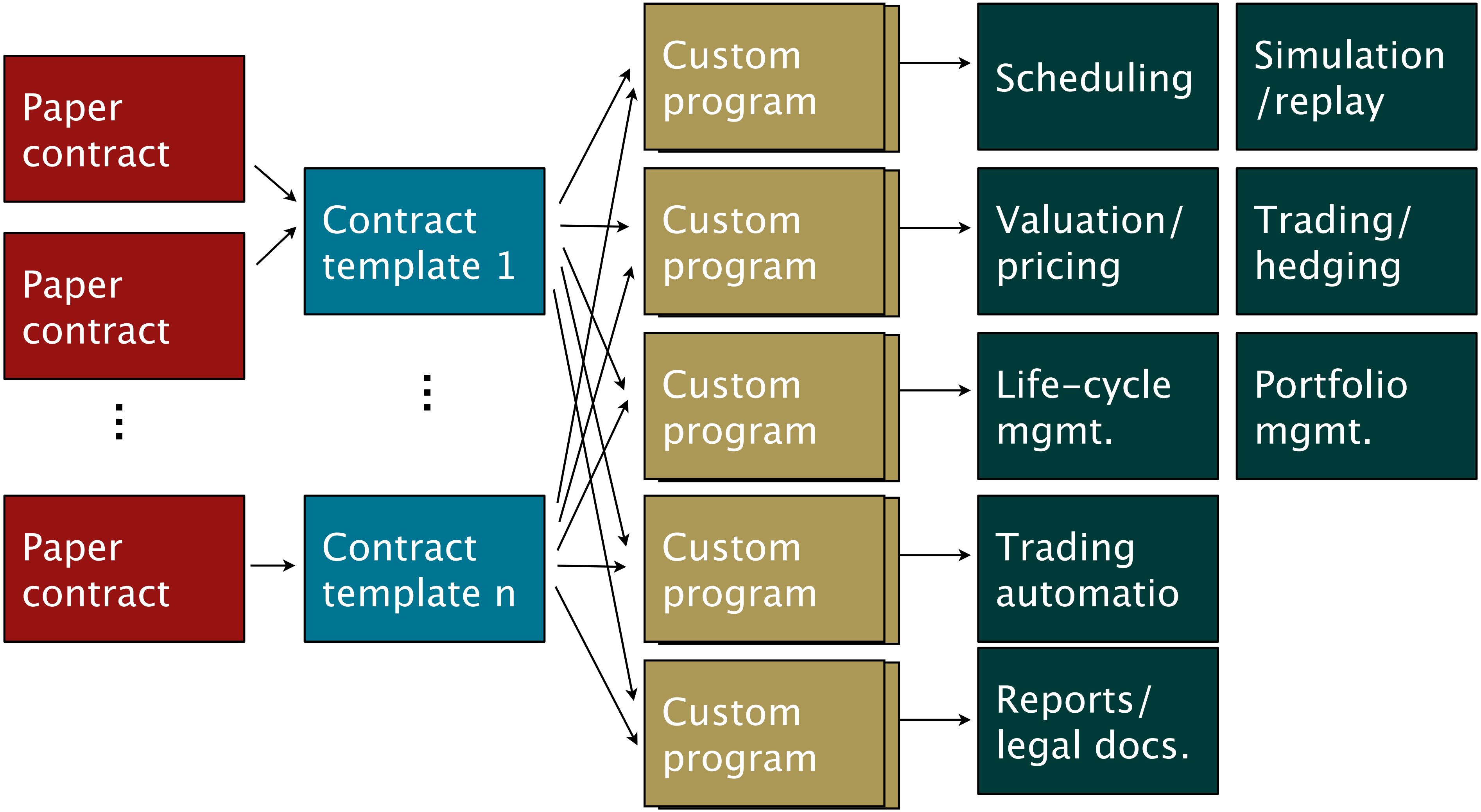
Contract modeling today

- Non-existent (paper-based)
  - Manual valuation/risk analysis is error-prone and slow
  - Easy to miss deadlines and opportunities
- Ad hoc/systematic, but directly coded
  - Pricing, scheduling, etc. must be coded for each new instrument. No way to verify code correctness.

Contract modeling today

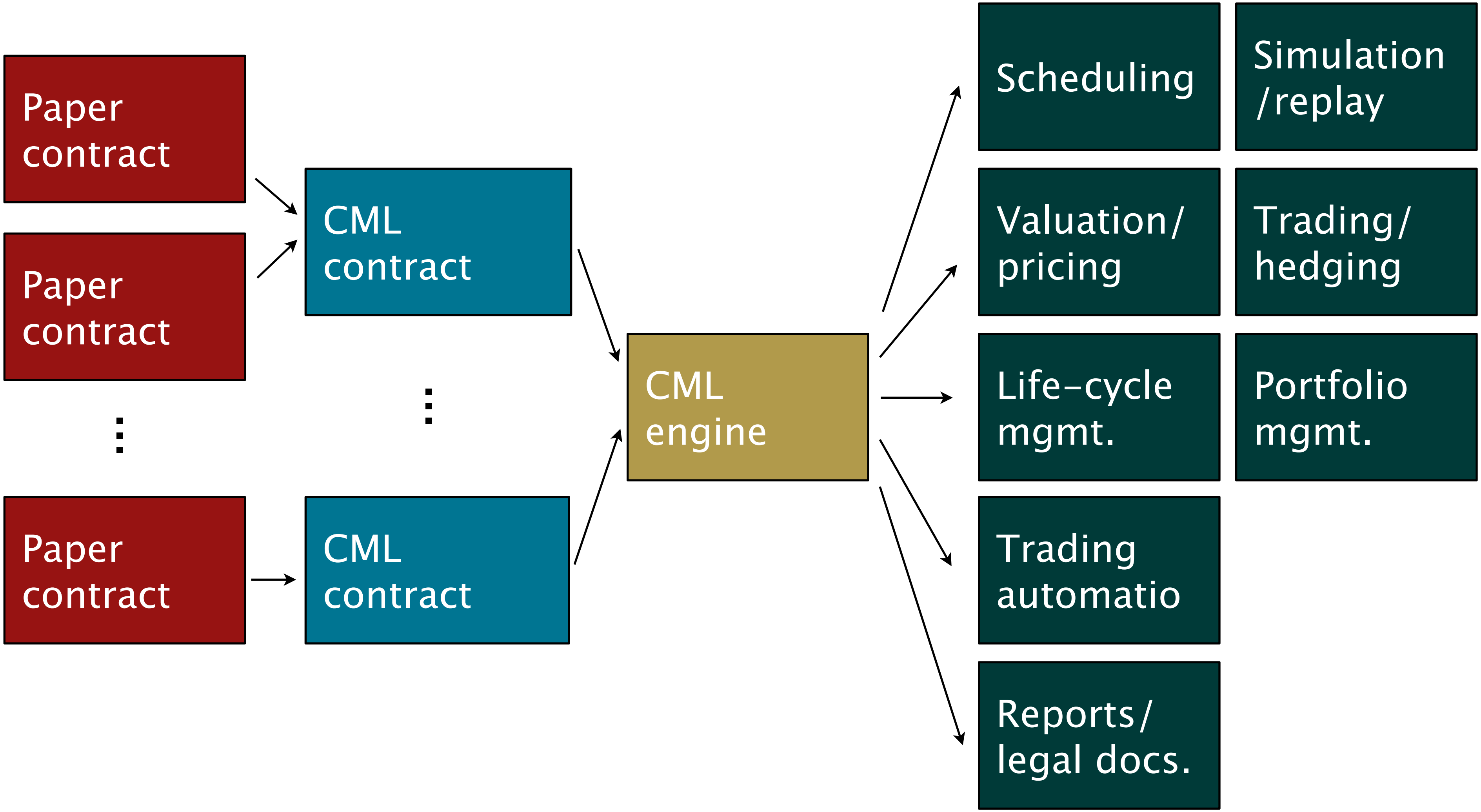
- Non-existent (paper-based)
  - Manual valuation/risk analysis is error-prone and slow
  - Easy to miss deadlines and opportunities
- Ad hoc/systematic, but directly coded
  - Pricing, scheduling, etc. must be coded for each new instrument. No way to verify code correctness.
- Using commercial platform
  - Fixed set of instruments – adding new types is costly
  - Integration is difficult (no standard representation)

Typical architecture



Montages tm
faster development

ContractML architecture


Paper contract

Paper contract

⋮

Paper contract

CML contract

⋮

CML contract

CML engine

Scheduling

Simulation /replay

Valuation/ pricing

Trading/ hedging

Life-cycle mgmt.

Portfolio mgmt.

Trading automatio

Reports/ legal docs.

Advantages of ContractML

- Programming contracts is less error–prone
- Pricing, scheduling, etc. require no extra coding
- Carry out all tasks on ongoing contracts too without any “custom programs” [new feature]
- Regulatory requirements easier to check (check once only!) [new feature]
- One less manual translation step makes many types of errors impossible

New perspectives

- Checking that business processes comply to contracts
- Formalizing SLAs (Service Level Agreements) to support knowledge workers and guarantee continuous compliance
- Simulation and replay
- Autonomous trading agents (electronic markets demand immediate action when price fluctuates)

Key Business Benefits

	Financial industry	Insurance companies	Others	Benefits
Scheduling	Easier	Easier	Easier	Op. risk ↓
Pricing (valuation, VaR)	Easier	Easier	Can do this now	Credit risk ↓
Integration/deal-capturing	Easier	Easier	Can do this now	Op. costs ↓
Autonomous trading agents	More is possible	More is possible	Can do this now	Op costs ↓
Legal description	Easier	Easier	Can do this now	Legal risk ↓
Simulation	More is possible	More is possible	Can do this now	Competitiveness ↑

Does your company systematically meet its contractual obligations?

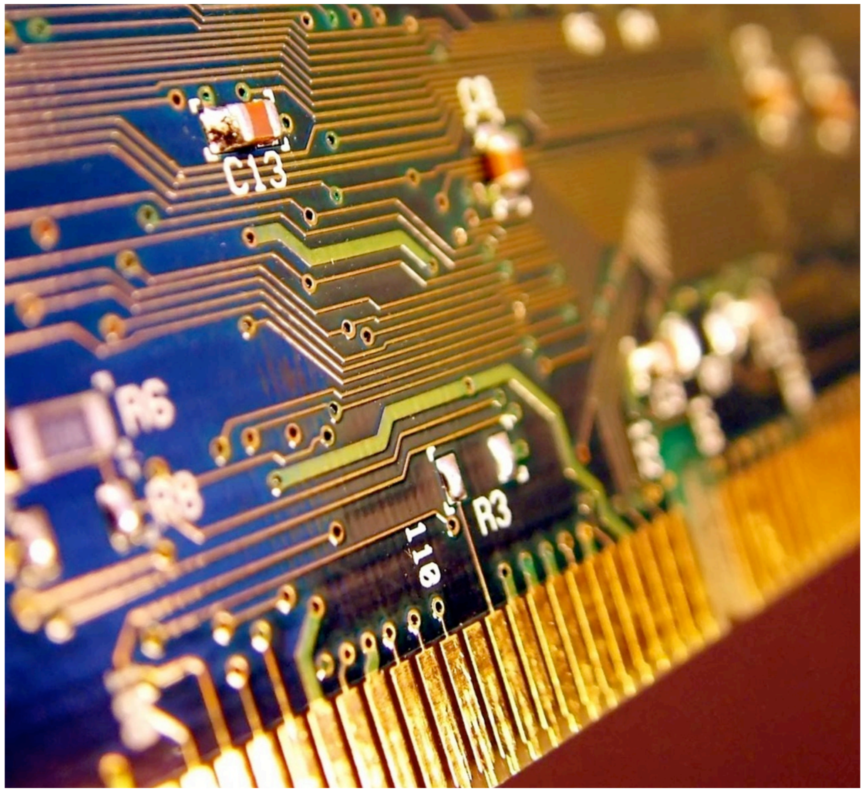
Yes, scheduling is now automatic even for new instruments.

Does your company loose money due to missed financial opportunities?

Valuation is now continuous and requires no extra coding.

Can you exchange contract information seamlessly between front- and back-office?

Yes, the standard representation ensures this.



The technology

Montages™
faster development

ContractML

ContractML

- Based on a few simple constructs:
  - Atomic contracts (**`transmit`**, **`success`**, **`fail`**)
  - Combinators (and, or, sequence)
  - Contract template declaration and invocation

ContractML

- Based on a few simple constructs:
  - Atomic contracts (**`transmit`**, **`success`**, **`fail`**)
  - Combinators (and, or, sequence)
  - Contract template declaration and invocation
- Compositional:
  - Simple contracts can be combined in a well-defined way to form more and more complex contracts.

Tested on 15+ contracts

Goods sale	Sale with installments
General contract	Agreement to sell
Balloon note	Contractor agreement
Legal services agreement	Danish trade law
Website development contract	Lease contract
Loan and security agreement	License agreement
Operating agreement (SLA)	Supply agreement
European option	Manufacturing agreement
American option	

Atomic contracts

Atomic contracts

- **`success`**

  No obligations, all agents are happy

Atomic contracts

- **`success`**

  No obligations, all agents are happy

- **`fail`**

  Breach of contract

Atomic contracts

- **`success`**

  No obligations, all agents are happy

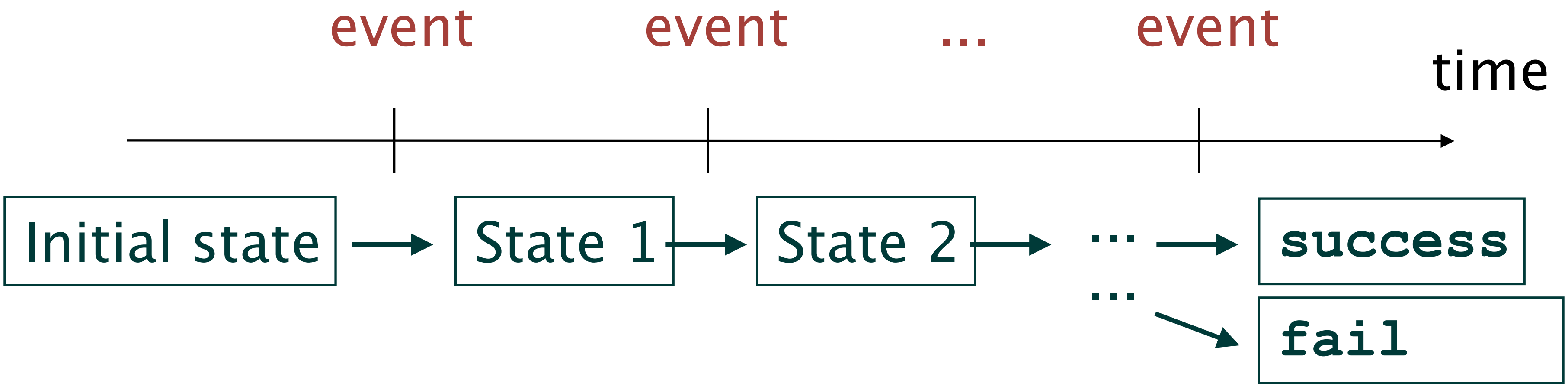
- **`fail`**

  Breach of contract

- **`transmit(sender,receiver,asset,condition)`**

  Obligates sender to transmit asset to receiver subject to the condition (usually a deadline). Sender has the initiative.

Evolving a contract


event

event

...

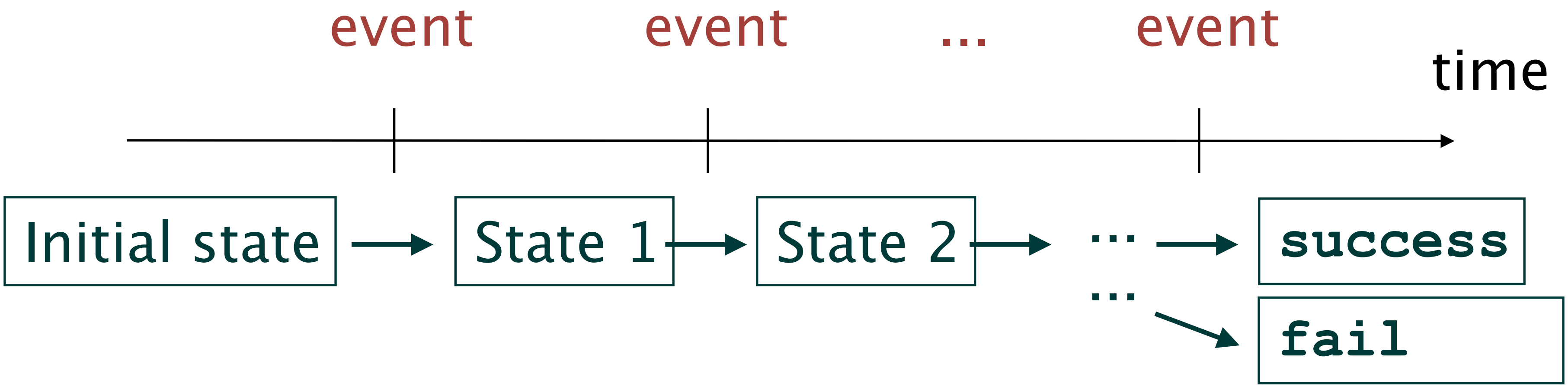
event

time

Initial state → State 1 → State 2 → ... → success

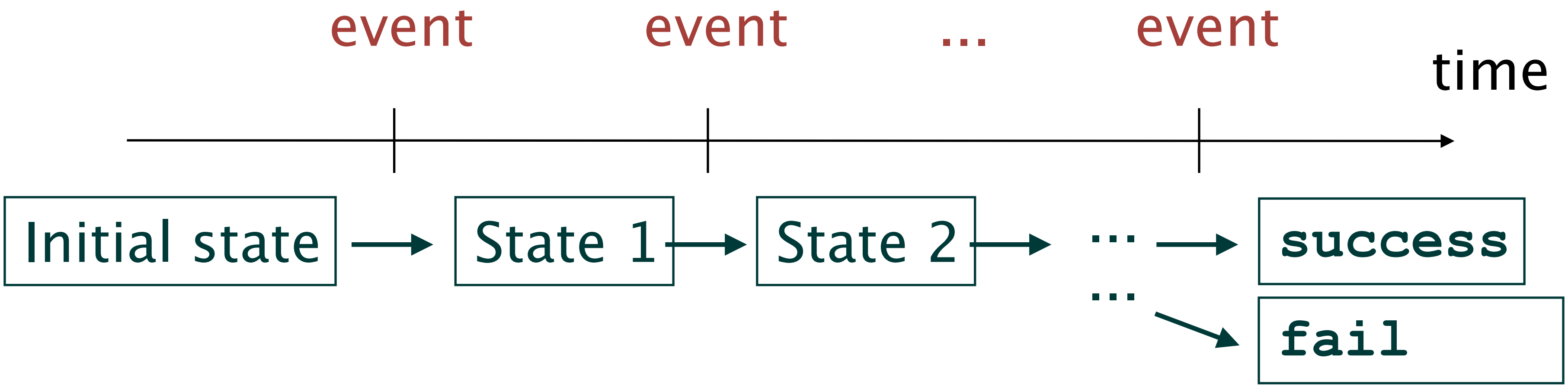
... → fail

Evolving a contract



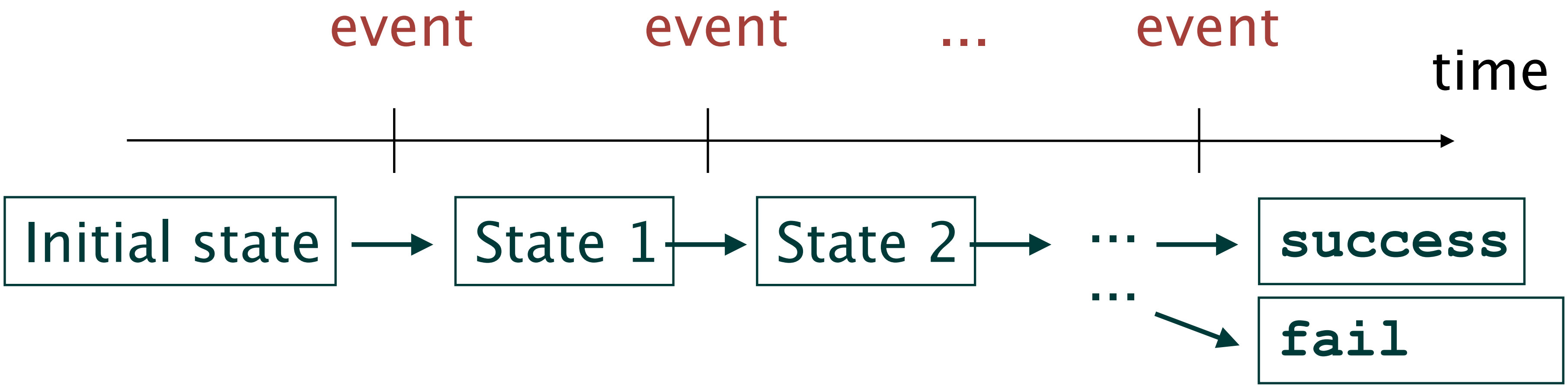
- Contract evolves from one state to another and ultimately become **`success`** or **`fail`**

Evolving a contract



- Contract evolves from one state to another and ultimately become **`success`** or **`fail`**
- Every event is a transmit event or a timer event

Evolving a contract



- Contract evolves from one state to another and ultimately become **`success`** or **`fail`**
- Every event is a transmit event or a timer event
- At any point in time the state of the system is the contract state plus the history of events.

Example: American option

Example:
American option

1. On or before <day> the holder <holder> may choose to acquire <underlying asset> at price <price> by remitting this amount to <issuer>. Issuer must transfer <underlying asset> to holder on the same day.

Example: American option

1. On or before <day> the holder <holder> may choose to acquire <underlying asset> at price <price> by remitting this amount to <issuer>. Issuer must transfer <underlying asset> to holder on the same day.
2. Should the holder choose not to exercise the option on or before <day>, this contract is void.

Example:
American option

1. On or before <day> the holder <holder> may choose to acquire <underlying asset> at price <price> by remitting this amount to <issuer>. Issuer must transfer <underlying asset> to holder on the same day.

2. Should the holder choose not to exercise the option on or before <day>, this contract is void.

3. If the paid amount is not received, insufficient or delayed for any reason, the holder looses the right to acquire <underlying asset> at said price.

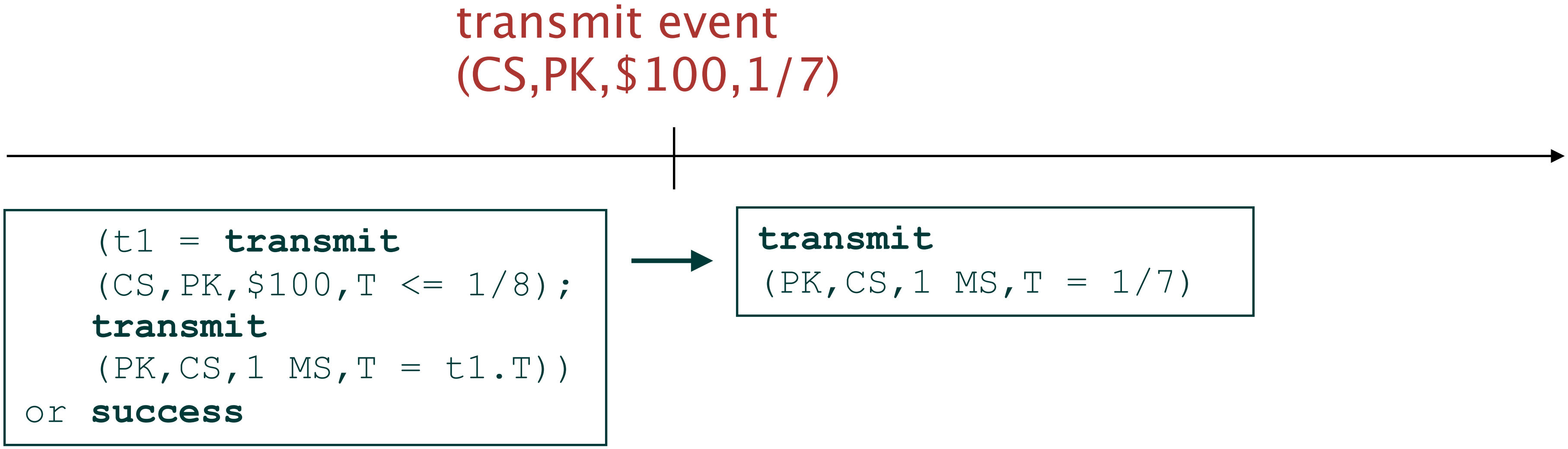
American option in ContractML

```
let
 usOption(issuer,holder,price,day,asset) =
    (t1 = transmit(holder,issuer,price,T <= day)
    ;transmit(issuer,holder,asset,T = t1.T))
  or success
in
  usOption(PK, CS, $100, 1/8, 1 MS)
end
```

Evolving the American option

```
    (t1 = transmit
    (CS,PK,$100,T <= 1/8);
   transmit
    (PK,CS,1 MS,T = t1.T))
or success
```

Evolving the American option

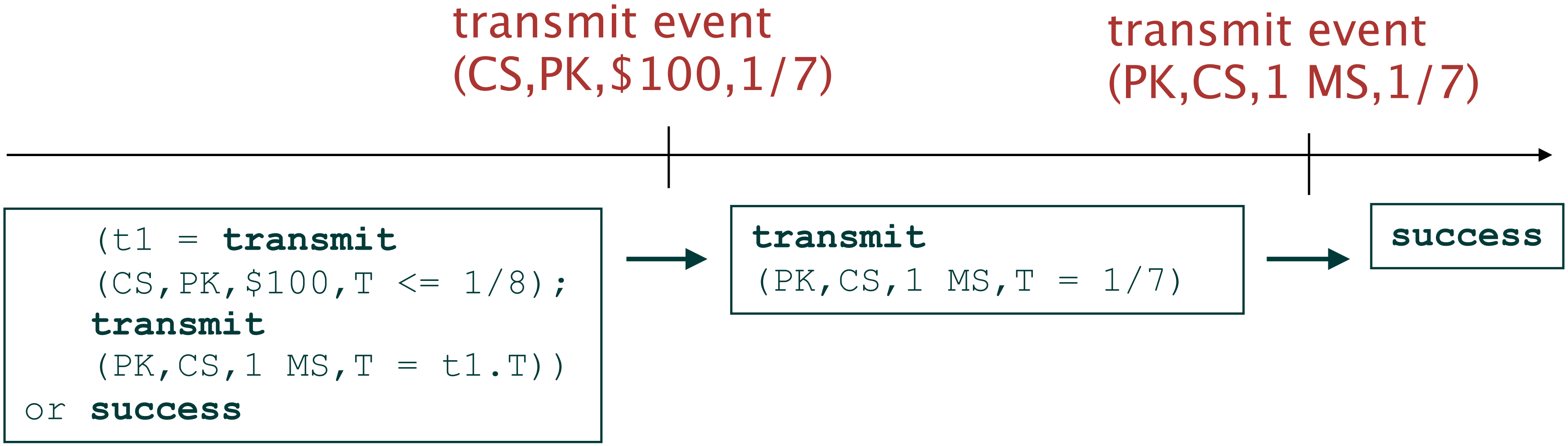


transmit event
(CS,PK,$100,1/7)

```
    (t1 = transmit
    (CS,PK,$100,T <= 1/8);
    transmit
    (PK,CS,1 MS,T = t1.T))
or  success
```

```
transmit
(PK,CS,1 MS,T = 1/7)
```

Evolving the American option



transmit event (CS,PK,$100,1/7)

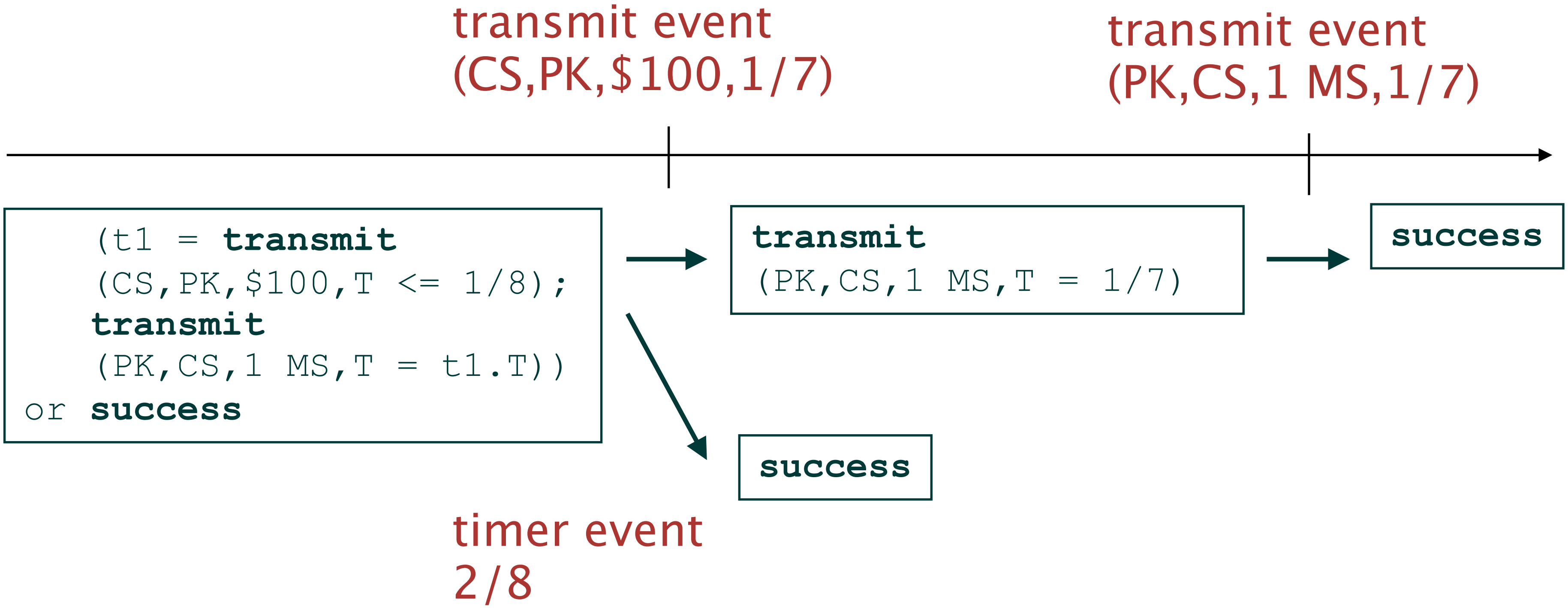
transmit event (PK,CS,1 MS,1/7)

```
    (t1 = transmit
    (CS,PK,$100,T <= 1/8);
    transmit
    (PK,CS,1 MS,T = t1.T))
or  success
```

```
transmit
(PK,CS,1 MS,T = 1/7)
```

```
success
```

Evolving the American option



transmit event
(CS,PK,$100,1/7)

transmit event
(PK,CS,1 MS,1/7)

```
    (t1 = transmit
    (CS,PK,$100,T <= 1/8);
   transmit
    (PK,CS,1 MS,T = t1.T))
or success
```

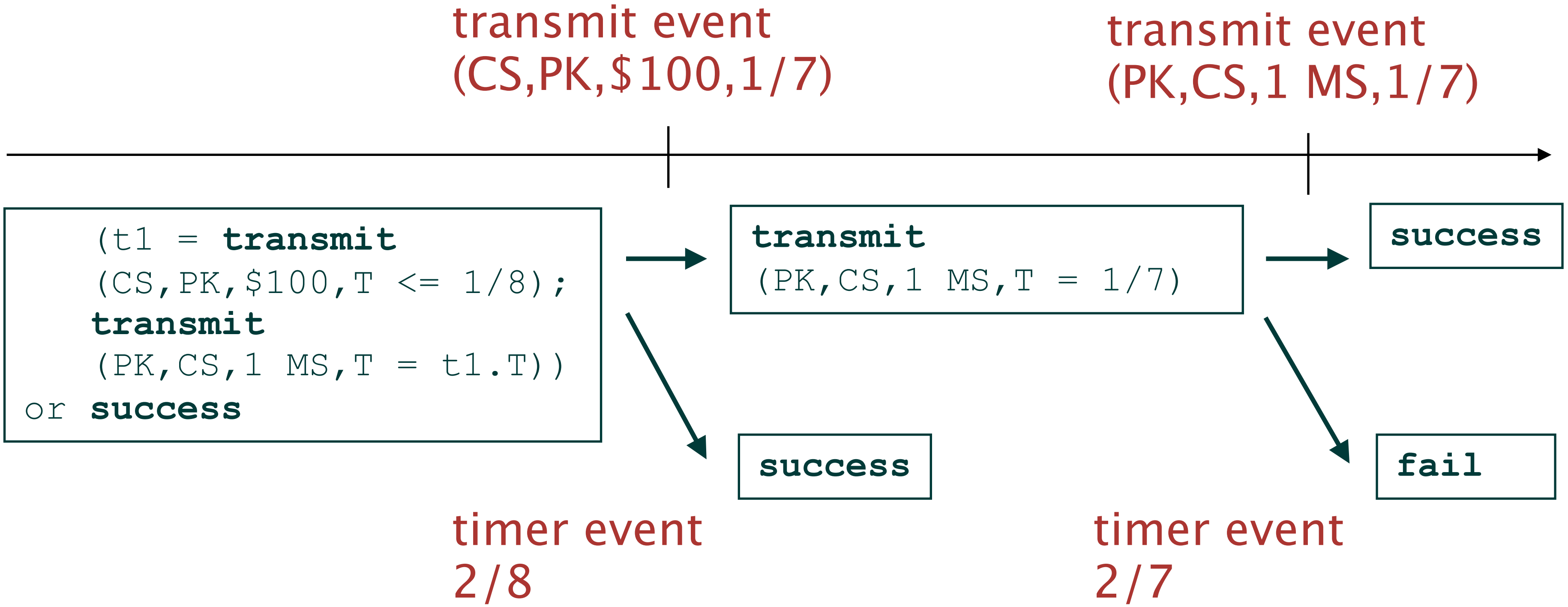
```
transmit
(PK,CS,1 MS,T = 1/7)
```

```
success
```

```
success
```

timer event
2/8

Evolving the American option



transmit event
(CS,PK,$100,1/7)

transmit event
(PK,CS,1 MS,1/7)

```
    (t1 = transmit
    (CS,PK,$100,T <= 1/8);
    transmit
    (PK,CS,1 MS,T = t1.T))
or  success
```

```
transmit
(PK,CS,1 MS,T = 1/7)
```

```
success
```

```
success
```

```
fail
```

timer event
2/8

timer event
2/7

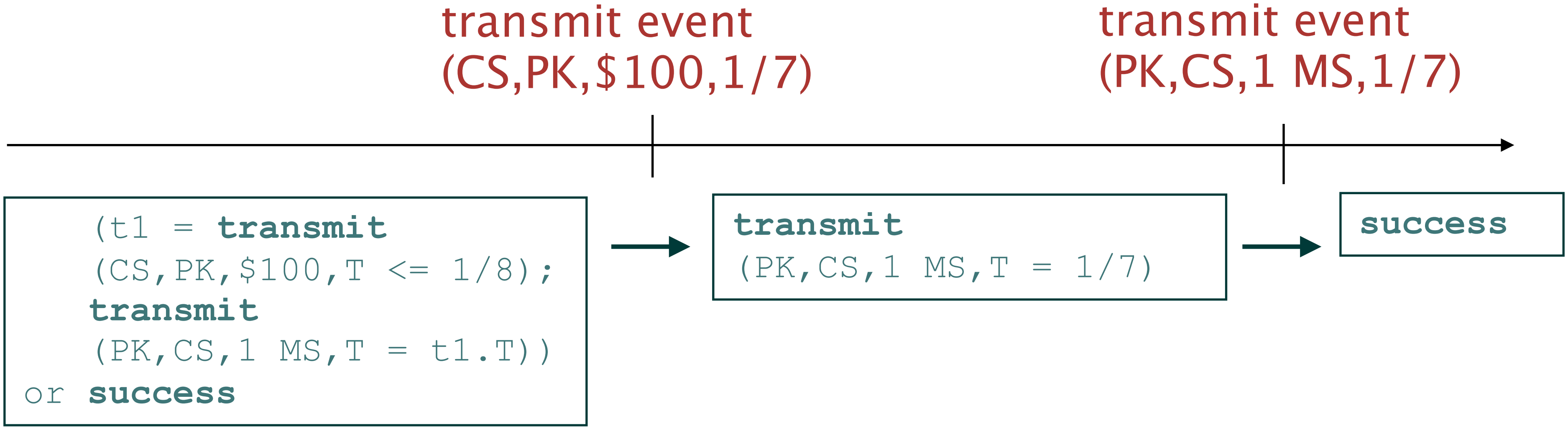
Why do we use DSLs?

– If the DSL is carefully designed, DSL programs can not only be run, but also analyzed – even while running

...and of course we have the usual benefits:

- Higher level of abstraction
- Less error-prone
- Etc.

Continuous analysis: scheduling/ valuation



transmit event (CS,PK,$100,1/7)

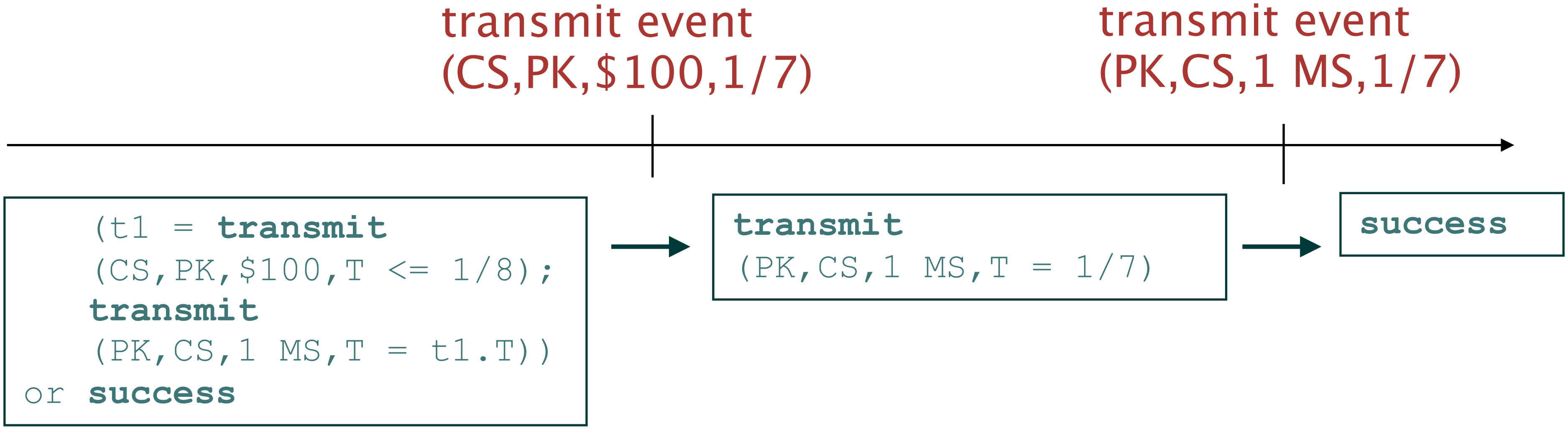
transmit event (PK,CS,1 MS,1/7)

```
    (t1 = transmit
    (CS,PK,$100,T <= 1/8);
    transmit
    (PK,CS,1 MS,T = t1.T))
or success
```

```
transmit
(PK,CS,1 MS,T = 1/7)
```

```
success
```

Continuous analysis: scheduling/ valuation



transmit event (CS,PK,$100,1/7)

transmit event (PK,CS,1 MS,1/7)

```
    (t1 = transmit
    (CS,PK,$100,T <= 1/8);
    transmit
    (PK,CS,1 MS,T = t1.T))
or success
```

```
transmit
(PK,CS,1 MS,T = 1/7)
```

```
success
```

Estimated current value: $x

Rights:

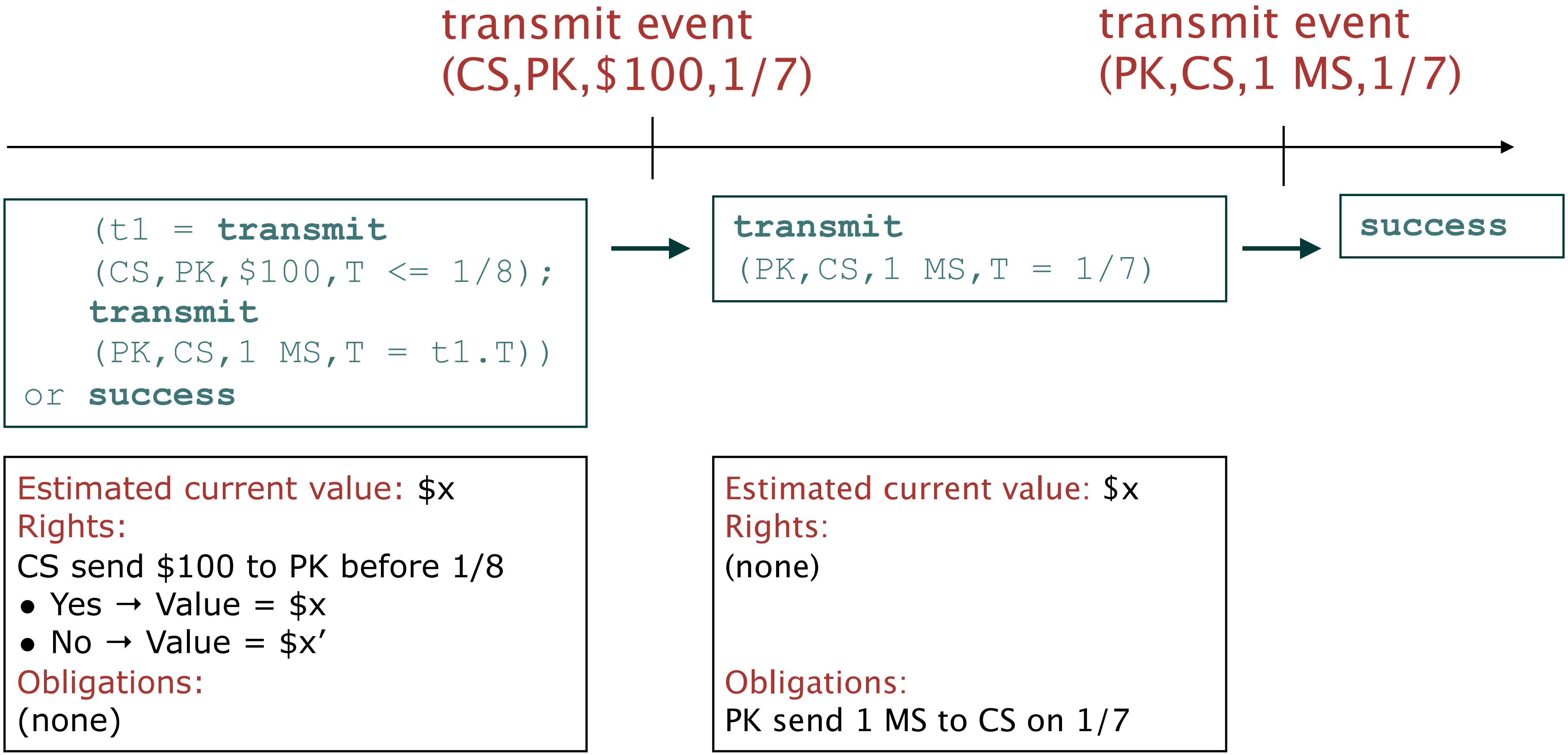
CS send $100 to PK before 1/8

- Yes → Value = $x
- No → Value = $x’

Obligations:

(none)

Continuous analysis: scheduling/ valuation



transmit event (CS,PK,$100,1/7)

transmit event (PK,CS,1 MS,1/7)

```
    (t1 = transmit
    (CS,PK,$100,T <= 1/8);
    transmit
    (PK,CS,1 MS,T = t1.T))
or  success
```

```
transmit
 (PK,CS,1 MS,T = 1/7)
```

```
success
```

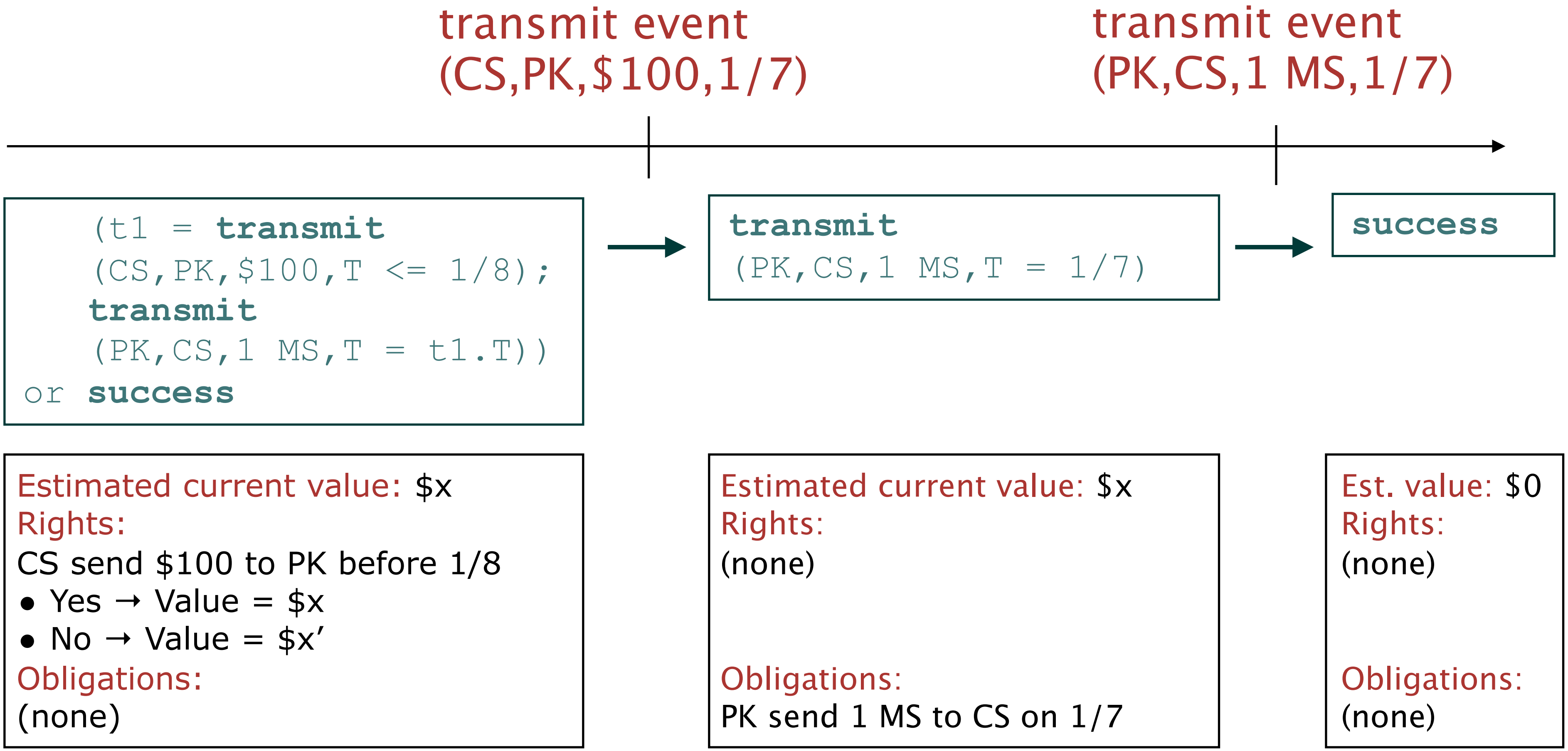
Estimated current value: $x
Rights:
CS send $100 to PK before 1/8
- Yes → Value = $x
- No → Value = $x’

Obligations:
(none)

Estimated current value: $x
Rights:
(none)

Obligations:
PK send 1 MS to CS on 1/7

Continuous analysis: scheduling/ valuation



transmit event (CS,PK,$100,1/7)

transmit event (PK,CS,1 MS,1/7)

```
    (t1 = transmit
    (CS,PK,$100,T <= 1/8);
    transmit
    (PK,CS,1 MS,T = t1.T))
or success
```

```
transmit
 (PK,CS,1 MS,T = 1/7)
```

```
success
```

Estimated current value: $x
Rights:
CS send $100 to PK before 1/8
- Yes → Value = $x
- No → Value = $x’

Obligations:
(none)

Estimated current value: $x
Rights:
(none)

Obligations:
PK send 1 MS to CS on 1/7

Est. value: $0
Rights:
(none)

Obligations:
(none)

Legal document generation

- usOption(holder, issuer, price, day, asset) =
- (t1 = **transmit** (holder,issuer, price,T <= day); **transmit**(issuer, holder,asset, T=t1.T))
- + **success**

- American option: *holder issuer, price, day, asset*
  Either <holder> can transmit to <issuer> the amount <price> no later than <day> and then <issuer> must transmit <asset> the same day.
- or the contract is complete and no rights or obligations remain.

Distinguishing features

	Contract ML	MLFi	FpML	Directly coded
Semantics	Many	Many	None	Few
Multi-partner	✓	×	✓	(✓)
Contract separate from analysis task	✓	✓	×	(×)
Analyze ongoing contracts	✓	✓	(×)	(×)
Independent agent/ resource model	✓	×	×	(✓)

Case studies

Montages tm
faster development

LexiFi / Société Générale

- Contract language MLFi with about 15 constructs
- Language description is publicly available.
- Handled all exotic options at Société Générale Asset Management
- Now made into a product and sold by LexiFi
- Constructs superseded by ContractML

Crédit Suisse
Global Modelling and Analytics Group

- 100.000+ derivative trades,
  including many exotic derivatives
- Needed daily updates to capital-at-risk, sensitivity, portfolio valuations, etc.
- Before, models were written in Excel
- Implemented DSL and analytics in Haskell
- → Stable, fewer errors, faster development

Jane Street Capital

- Proprietary New York–based trading firm
- Implemented all trading/analytics systems in OCaml.
- Get correctness guarantees that are essential to financial systems.
- High–level executives can (and do) review the code!

J.P. Morgan Kapital

Axel Kramer

- Middle office system
- DSL for financial instruments using valuation-independent financial event templates
- Mark to market and sensitivity are the most important analyses
- Was granted U.S. patent (#127341)
- → Increased profit because exotics could be brought to the market faster

Cap Gemini

Arie van Deursen

- Banks frequently invent new financial products and need them to be understood by automated systems.
- Solution compiles DSL contracts descriptions to legacy formats and Cobol programs
- Included in their Financial Product System (FPS) and used in several Dutch banks
- Stopped selling the system for unknown reasons

Others of interest

- HypoVereinsbank, München
  Exotic equity derivatives in Scheme 48
  Michael Sperber
- ABN AMRO
  Counterparty risk on financial derivatives
  Cyril Schmidt
- See academic work by Henglein, Peyton Jones or Prisacariu
- Also see the annual CUFP workshop

Moving forward

- Make prototype ready for demoing (Philipp Kutter / Christian Stefansen)
- Identify test customer to drive requirements
- Strengthen business case

www.stefansen.dk