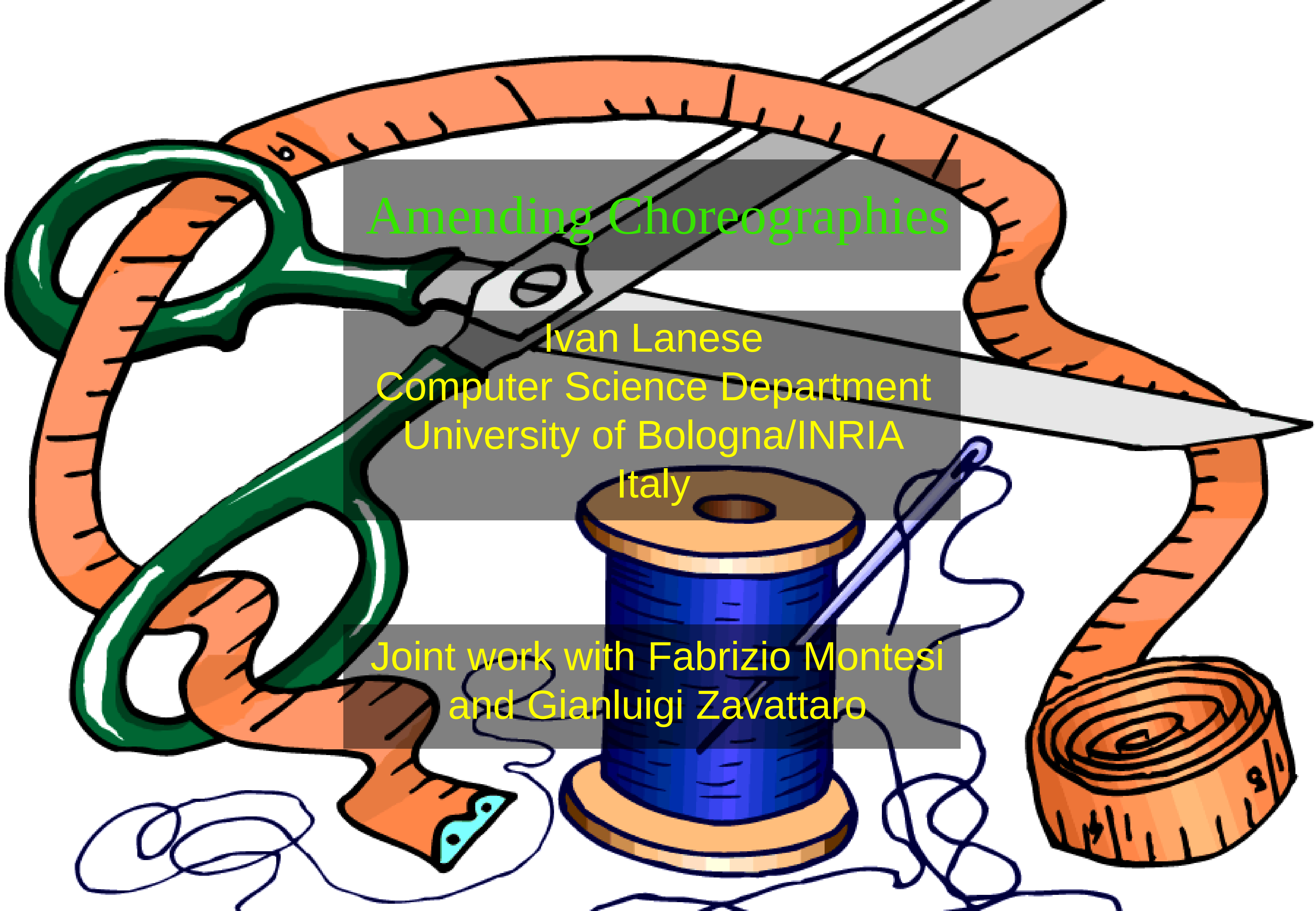


Amending Choreographies

Ivan Lanese
Computer Science Department
University of Bologna/INRIA
Italy

Joint work with Fabrizio Montesi
and Gianluigi Zavattaro

Map of the talk

- Choreographies
- Amending choreographies
- Conclusions

Map of the talk

- Choreographies
- Amending choreographies
- Conclusions

Choreographies

- Allow to describe the behavior of a distributed communicating system at the very abstract level
- Composed by interactions of the form a $\rightarrow b: o$
- Using different operators
  - Sequential composition ;
  - Parallel composition ||
  - Nondeterministic choice +
- There are approaches extending choreographies with additional information (data, recursion, ...)
- Very similar to global types in multiparty session types

Choreography projection

- Allows to automatically derive from a choreography the description of the behavior of each participant
- Nearer to the implementation
- Preserves the semantics: when interacting, the participants behave as specified by the choreography

Participants description

- Locations corresponding to participants, containing their code
- Basic operations: input o and output $\bar{o}$
- Composed using
  - sequential composition ;
  - parallel composition |
  - nondeterministic choice +

Examples

- The projection of a $\rightarrow b$: o; b $\rightarrow c: o'$ is $[\bar{o}]_a \mid\mid [o; \overline{o'}]_b \mid\mid [o']_c$
- The projection of a $\rightarrow b$: o; $c \rightarrow d: o'$ is $[\bar{o}]_a \mid\mid [o]_b \mid\mid [\overline{o'}]_c \mid\mid [o']_d$
  - Not well-behaved

Well-behaved choreographies

- Syntactic conditions ensure choreographies are well-behaved
- Conditions depend on
  - Synchronous or asynchronous semantics
  - For asynchronous, whether send, receive or both are observed

Non well-behaved choreographies

- What to do when choreographies are not well-behaved?
- We transform them automatically into well-behaved ones
  - According to the strictest of the conditions
- Preserving the intended semantics
  - Weak traces

Map of the talk

- Choreographies
- Amending choreographies
- Conclusions

Choreography issues

- Connectedness for sequence
  - Two roles do not agree on when the first term of a sequential composition has been completed
- Unique points of choice
  - Two roles do not agree on which branch of a choice has been taken
- Causality safety
  - A send on an operation is not received by the intended target, but by another receiver

Our approach

- For each issue we show a pattern for solving it
  - Preserving weak traces
- We compose all the patterns into a unique algorithm solving all the issues

transI and transF

- Auxiliary definition needed for the formalization
- transI: set of interactions enabled in a term
- transF: set of interactions that may be the last one to be executed in the term
- Can be defined by structural induction on the term

Connectedness for sequence issue

- $C = a \to b: o; c \to d: o'$
- $proj(C) = [\bar{o}]_a \mid\mid [o]_b \mid\mid [\overline{o'}]_c \mid\mid [o']_d$
- c does not know when b has received the message
- Avoided if for each subterm of the form C';C''
  - There is a participant that knows when C' ends which regulates when C'' starts
  - Formally, for each $a \to b: o$ $\in$ transF(C'), each $c \to d: o' \in$ transI(C'') we have $b = c$

Connectedness for sequence pattern

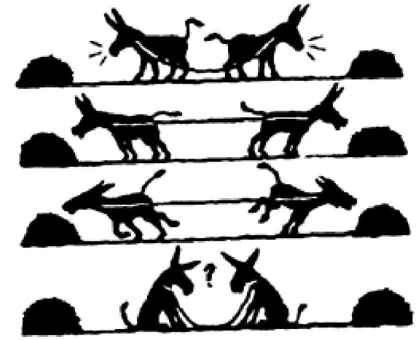
- Introduce a new coordinator role e
  - Checks when C' ends
  - Allows C'' to start
- Replace each $a \rightarrow b: o \ \in$ transF(C') with

$$a \rightarrow b: o; b \rightarrow e: o^{*1}$$

- Replace each c $\rightarrow d: o' \ \in$ transI(C'') with

  e $\rightarrow c: o^{*2}; c \rightarrow d: o'$
- Operations with * are private, not visible in weak traces

Unique points of choice issue (1)



- $C = a \rightarrow b{:}\, o + c \rightarrow d{:}\, o'$
- $proj(C) = [\bar{o} + 1]_a \;||\; [o + 1]_b \;||\; [1 + \overline{o'}]_c \;||\; [1 + o']_d$
- Avoided if for each subterm C’+C’’
  - There is a participant deciding which branch to take and notifying all the others
  - Formally, for each $a \rightarrow b{:}\, o$ ∈ transI(C’+C’’), a is the same
- Role a decides which branch to take

Unique points of choice issue (2)

- $C_1 = (a \rightarrow b: o + a \rightarrow c: o'); b \rightarrow c: o''$
- $proj(C_1) = [\bar{o} + \overline{o'}]_a \mid\mid \left[(o + 1); \overline{o''}\right]_b \mid\mid [(1 + o'); o'']_c$
- Avoided if for each subterm C’+C’’
  - C’ and C’’ have the same set of roles
- All the involved roles do something in each alternative
- No projection with one branch 1 and one branch non 1

Unique points of choice pattern

- Introduce a new coordinator role e making the choice
- Replace each $a \rightarrow b : o$ ∈ transI(C'+C'') with
  e $\rightarrow a: o^{*1}; a \rightarrow b: o$
- If role f occurs in C' but not in C'' then transform C'' into $\mathrm{C}''||e \rightarrow f: o^{*3}$
  - And vice versa

Causality safety issue

- Avoided if for each pair of interactions $a \rightarrow b\colon o$ and $c \rightarrow d\colon o$ on the same operation
  - If the send at a may trigger, the receive at d is not enabled
  - If the send at c may trigger, the receive at b is not enabled
- We need either a causal dependence or a conflict
- Different approaches depending on the top-level operator in the smallest term containing both the interactions

Sequential causality safety issue

- $C = a \to b : o ; b \to c : o$
- $proj(C) = [\bar{o}]_a \mid\mid [o ; \bar{o}]_b \mid\mid [o]_c$
- The send at a may be get by the receive at c
- May happen in terms C';C'' with interactions $a \to b : o$ in C' and c $\to d : o$ in C''
- If the term satisfies connectedness for sequence there is a dependence between the receive at b and the send at c
- We add the missing dependence by replacing c $\to d : o$ with

$$\mathrm{c} \to d : o^{*1} ; d \to c : o^{*2} ; \mathrm{c} \to d : o$$

Choice causality safety issue

- $C = (a \to b{:}\, o'; b \to a{:}\, o; a \to c{:}\, o'') +$
  $(a \to c{:}\, o; c \to b{:}\, o')$
- $proj(C) = [\overline{o'}; o; \overline{o''} + \bar{o}]_a \mid\mid [o'; \bar{o} + o']_b \mid\mid [o'' + o; \overline{o'}]_c$
- The message on o to a may be get by c
- The problem is that c has not been notified about the choice yet
- We add the missing dependence by replacing $a \to c{:}\, o$ with
  $a \to c{:}\, o^{*1}; c \to a{:}\, o^{*2}; a \to c{:}\, o$

Parallel causality safety

- $C = a \to b: o || c \to d: o$
- $proj(C) = [\bar{o}]_a \;||\; [o]_b \;||\; [\bar{o}]_c \;||\; [o]_d$
- The send at a may be get by the receive at d
- Parallel causality safety cannot be amended by adding (private) interactions only

Choreography normal form

- A choreography is in normal form if it is

$$\sum_i a_i \rightarrow b_i : o_i ; C_i$$

where C_i is in normal form

Expansion law

- We can use the expansion law to put choreographies in normal form

$$\left(\sum_i a_i \to b_i : o_i ; C_i\right) || \left(\sum_i a_j \to b_j : o_j ; C_j\right) =$$
$$\sum_i a_i \to b_i : o_i ; \left(C_i || \sum_i a_j \to b_j : o_j ; C_j\right) +$$
$$\sum_j a_j \to b_j : o_j ; \left(C_j || \sum_i a_i \to b_i : o_i ; C_i\right)$$

- This solves parallel causality safety issues

Amending choreographies

- We can compose the patterns above to transform any choreography into a well-behaved one
  1. Apply the pattern to solve parallel causality issues
  2. Apply the pattern for connectedness for sequence and unique points of choice
  3. Apply the pattern for sequential and choice causality safety
  - All patterns applied from smallest subterms to largest subterms

Final result

- The transformation preserves weak traces and makes the choreography well-behaved
  - For synchronous semantics
  - For asynchronous semantics, observing either send, or receive, or both
- The projection of a well-behaved choreography preserves traces
- The projection of the transformed choreography is weak trace equivalent to the original choreography

Map of the talk

- Choreographies
- Amending choreographies
- Conclusions

Summary

- An automatic technique for transforming a given choreography into a projectable one
- The transformation preserves weak traces
- All patterns but the one for parallel causality safety based on adding auxiliary interactions
- The pattern for parallel causality safety reduces the degree of concurrency
- Patterns are applied only when and where they are needed

Future work

- Extend the approach to deal with other features
  - Recursion
  - Data
- Exploiting choreography amending for choreography composition
  - Adaptive choreographies

End of talk

Thanks!

Questions?