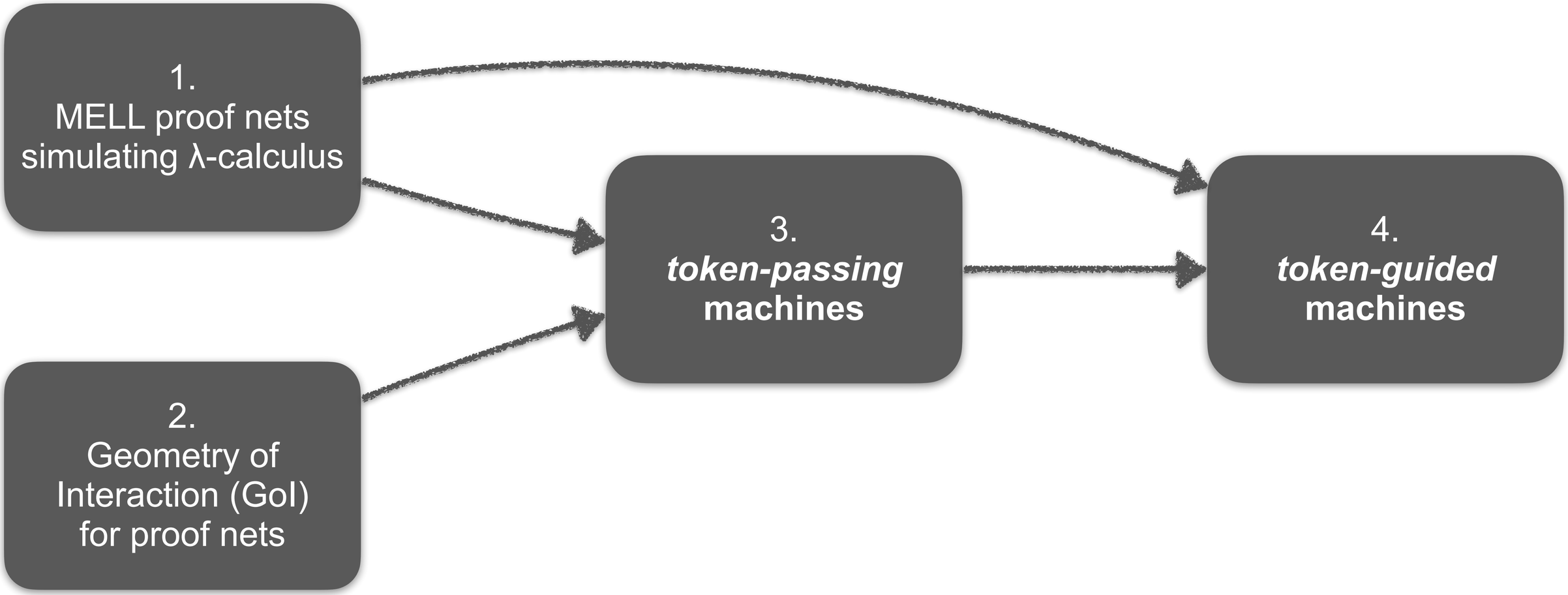
Program semantics with token passing

Koko Muroya
(RIMS, Kyoto University)

Linear Logic Winter School (Marseille / hybrid), 28 January 2022

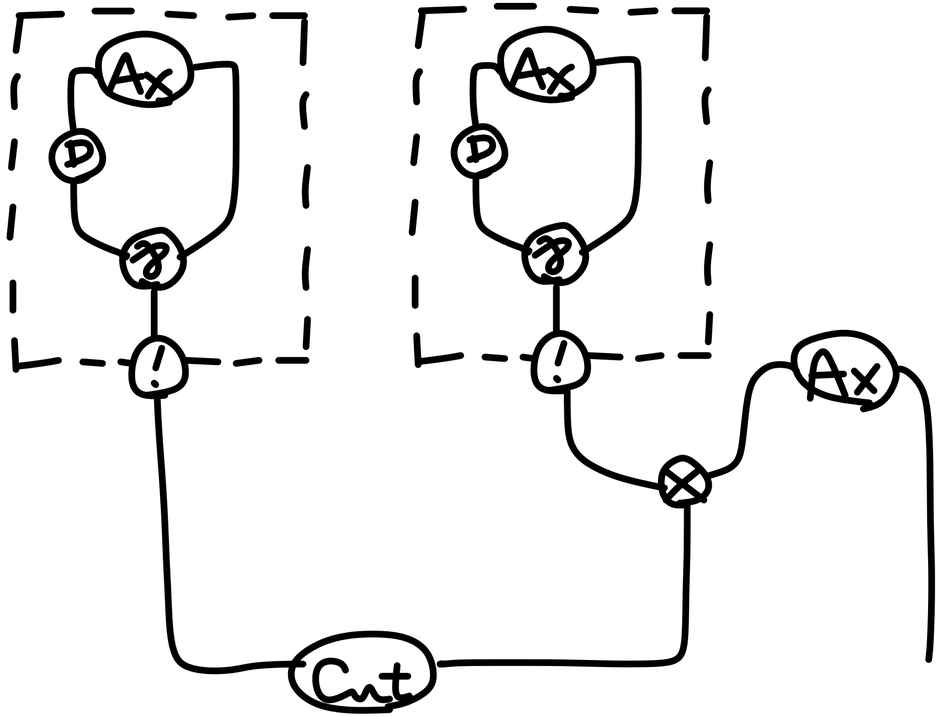
Program semantics with token passing: overview



1. MELL proof nets simulating λ-calculus

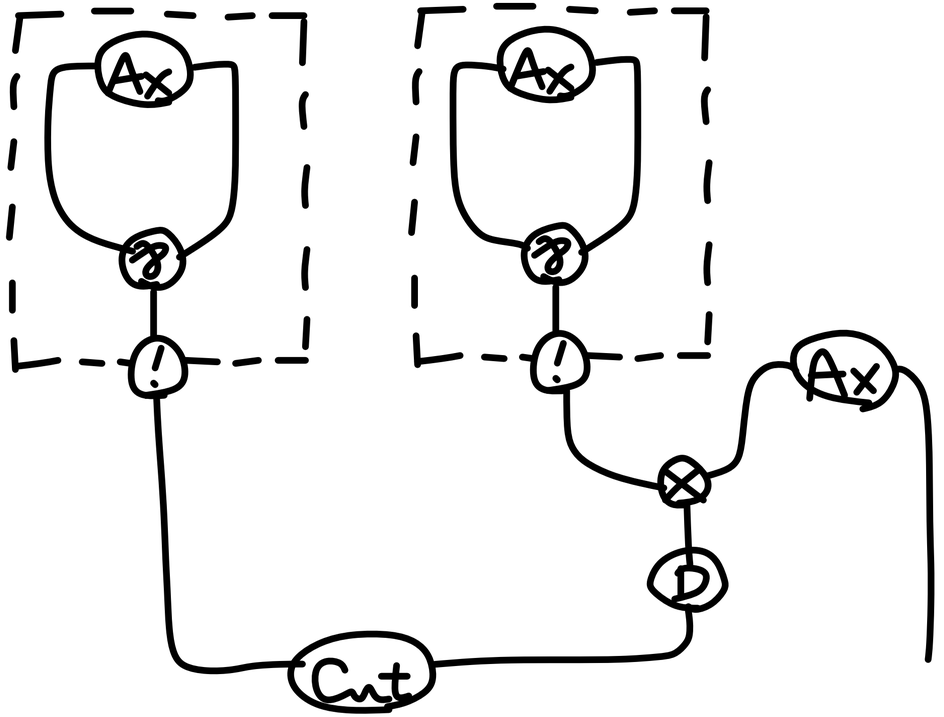
- “call-by-name” translation

$$(A \Rightarrow B) \mapsto (!A \multimap B)$$



- “call-by-value” translation

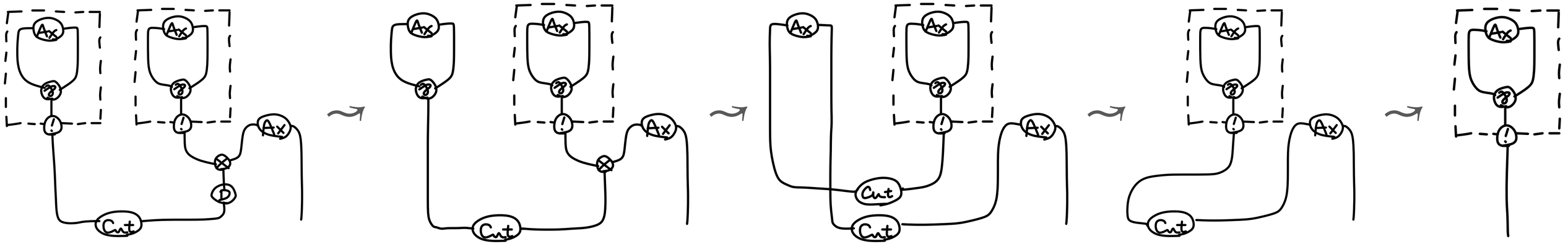
$$(A \Rightarrow B) \mapsto !(A \multimap B)$$



$$(\lambda x\,.\,x)\ (\lambda y\,.\,y)$$

1. MELL proof nets simulating λ-calculus

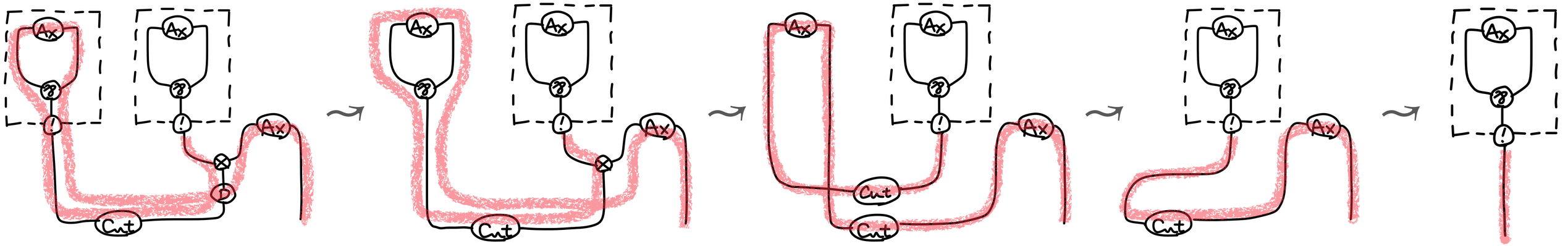
- “call-by-value” translation $(A \Rightarrow B) \mapsto !(A \multimap B)$
- cut elimination simulating β-reduction



$$(\lambda x\,.\,x)\ (\lambda y\,.\,y) \quad \to_\beta \quad \lambda y\,.\,y$$

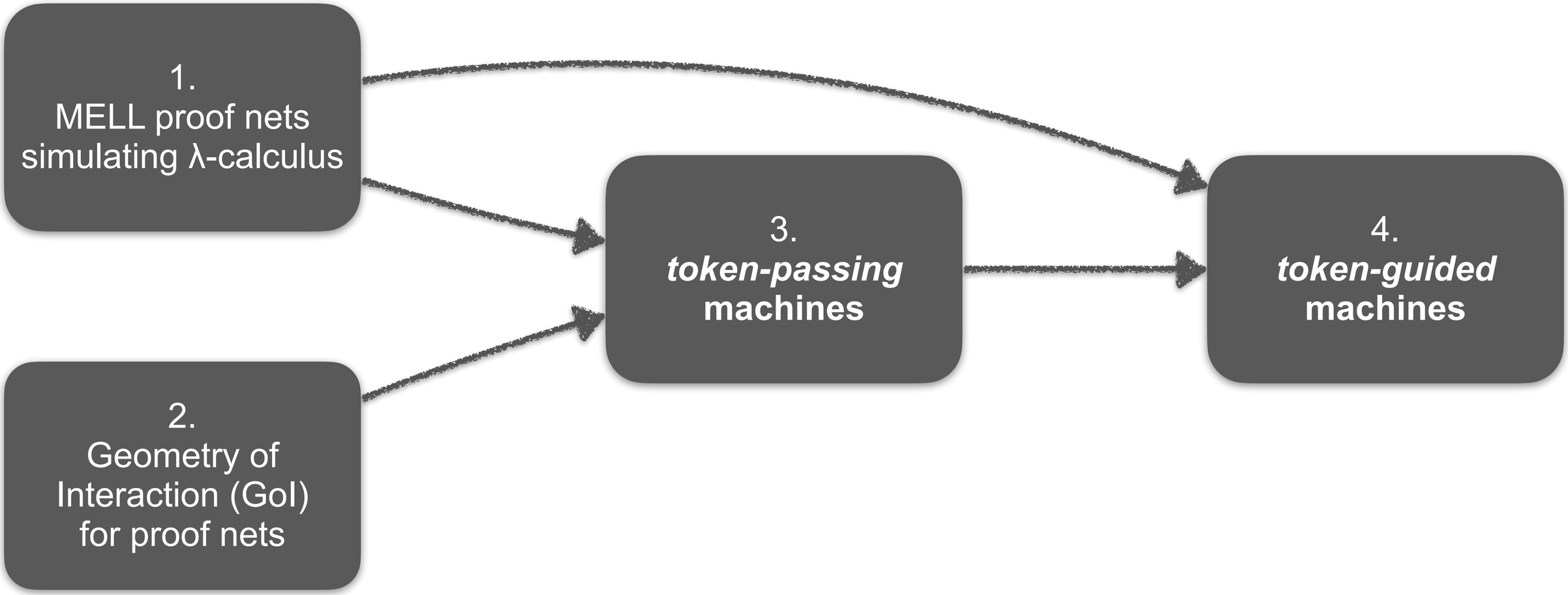
2. Geometry of Interaction (GoI) for proof nets

- invariant under cut elimination (and β-reduction), as *paths*



$$(\lambda x\,.\,x)\ (\lambda y\,.\,y) \quad \to_\beta \quad \lambda y\,.\,y$$

Program semantics with token passing: overview



2. *Token-passing* machines

- semantics of functional programs
  - *programs* represented graphically (typically as proof nets)
  - *evaluation* modelled by dynamic computation of the GoI invariant

***passing a token* around a fixed proof net**

- pioneering work, with application to compiler construction

  [Danos & Regnier, TCS 1999] [Mackie, POPL 1995]

2. *Token-passing* machines

- pioneering work

  [Danos & Regnier, TCS 1999]

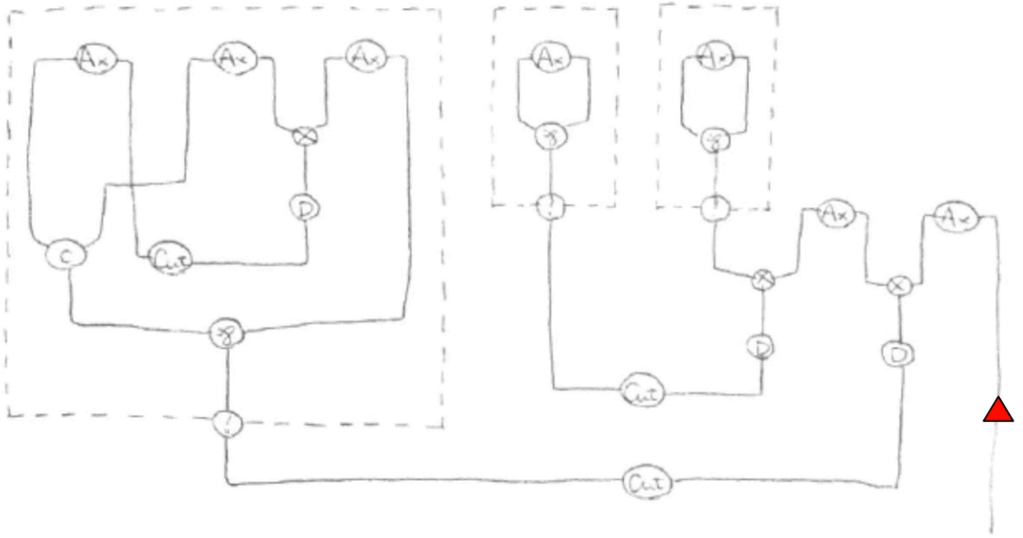
slides from CSL 2017 talk
https://www.kurims.kyoto-u.ac.jp/~kmuroya/talks/csl17.pdf

Interaction abstract machine (IAM)

$$(\lambda x.x\,x)\,((\lambda y.y)\,(\lambda z.z)) \;\Downarrow\; \lambda z.z$$

GoI token passing fixed graph

$(\lambda x.x\,x)\,((\lambda y.y)\,(\lambda z.z))$



2. *Token-passing* machines

***passing a token* around a fixed proof net**

- semantics of functional programs
  - *programs* represented graphically (typically as proof nets)
  - *evaluation* modelled by dynamic computation of the GoI invariant

- pioneering work, with application to compiler construction

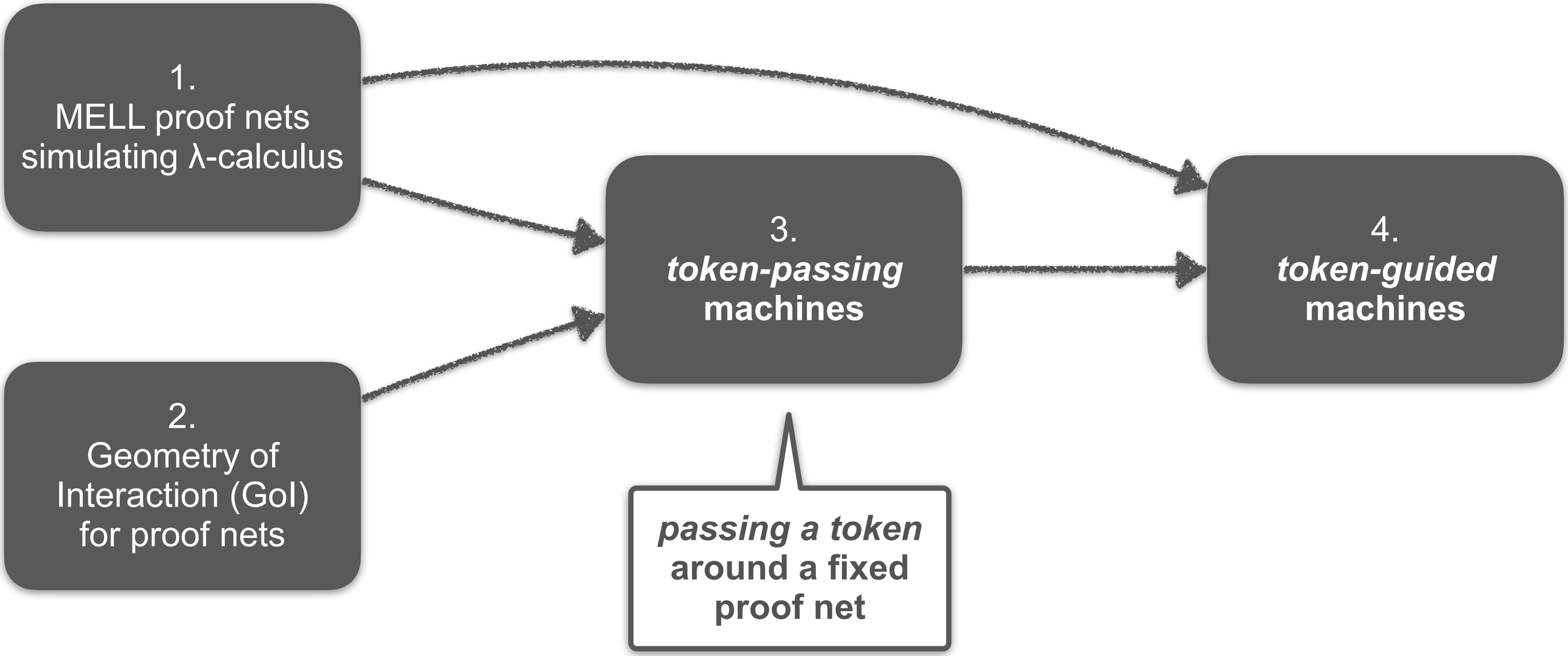
  [Danos & Regnier, TCS 1999] [Mackie, POPL 1995]

- variants for effectful programs

  [Schöpp, APLAS 2011] [Hasuo & Hoshino, LICS 2011] [Hoshino, M. & Hasuo, CSL-LICS 2014, with a follow-up paper in POPL 2016] [Dal Lago & Hoshino, LICS 2019]

- multi-token variants for effectful programs & parallelism [Dal Lago, Faggian, Hasuo & Yoshimizu, CSL-LICS 2014, with follow-up papers in LICS 2015, POPL 2017]
- application to high-level synthesis [Ghica, POPL 2007, with follow-up papers in ENTCS 2010, POPL 2011, ICFP 2011]

Program semantics with token passing: overview



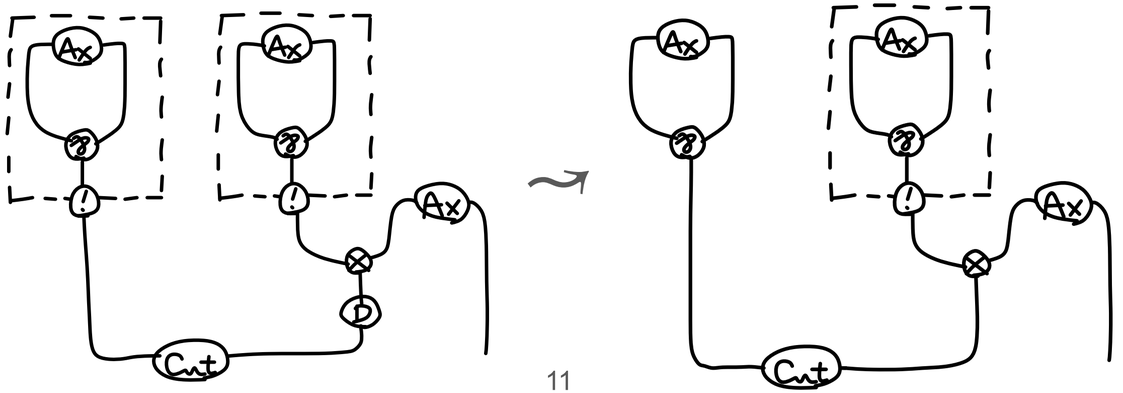
From *without* rewriting to *with* rewriting

- *token-passing* machines
  - dynamic computation of the GoI invariant *without* rewriting (cut elimination)

***passing a token* around a fixed proof net**

- Question:

  What happens if *dynamic* rewriting is allowed in token-passing machines?
  - Would that speed up token passing, by removing overlapping paths?



From *without* rewriting to *with* rewriting

- Question:

  What happens if *dynamic* rewriting is allowed in token-passing machines?

- Answer(s):

  - It is possible in a principled way, inspired by virtual reduction [Danos & Regnier, LICS 1993]

  - The token witnesses & controls possible rewriting!

From *without* rewriting to *with* rewriting

***passing a token* around a fixed proof net**

- Question:

  What happens if *dynamic* rewriting is allowed in token-passing machines?

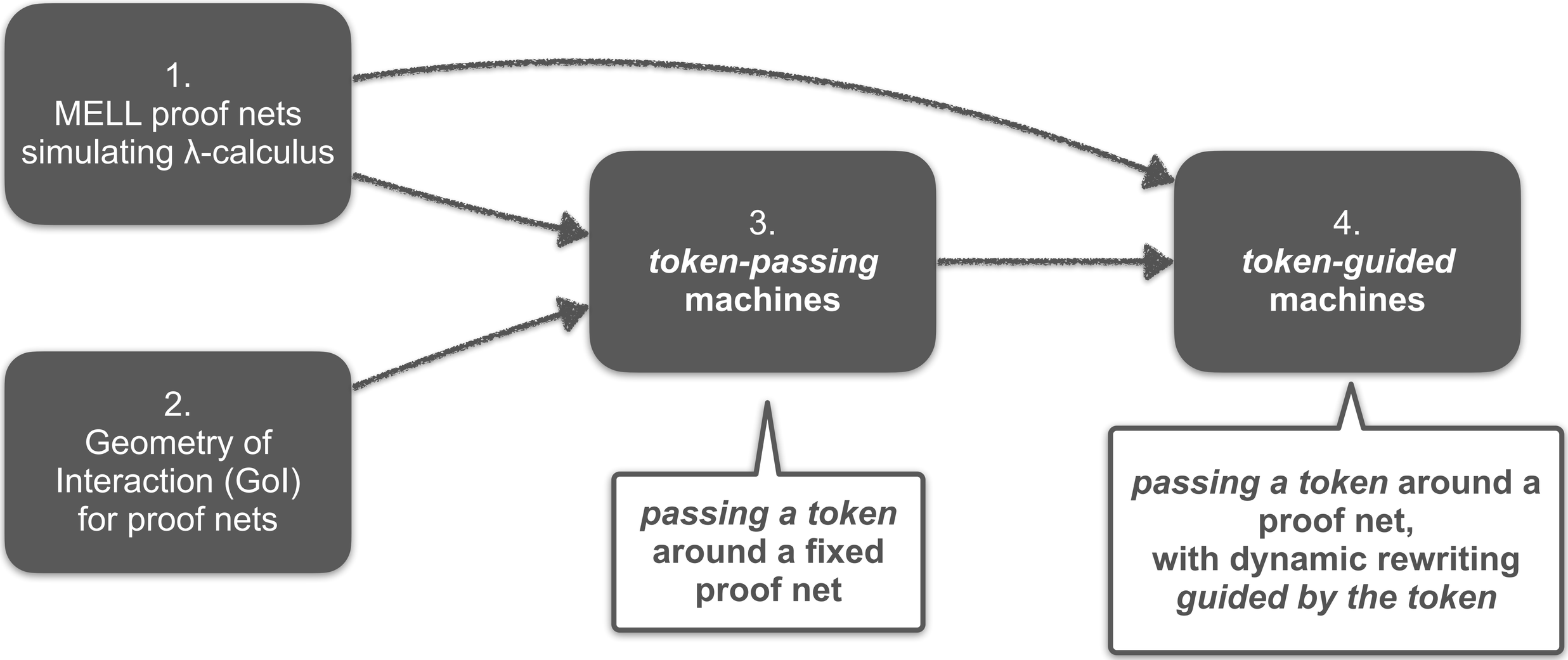
- Answer(s):

  - It is possible in a principled way, inspired by virtual reduction [Danos & Regnier, LICS 1993]
  - The token witnesses & controls possible rewriting!

→ *token-guided* machines

passing a token* around a proof net, with dynamic rewriting *guided by the token

Program semantics with token passing: overview



4. *Token-guided* machines

- semantics of functional programs
  - *programs* represented graphically
  - *evaluation* modelled by dynamic computation of the GoI invariant,
    together with dynamic rewriting of graphs
  - *evaluation strategies* represented by the token

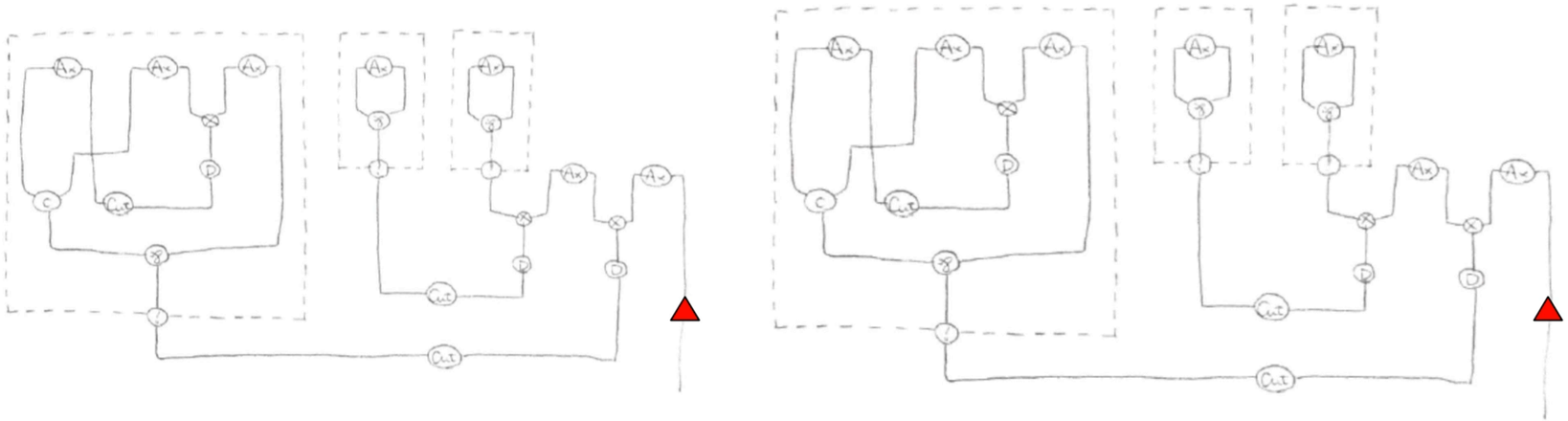
passing a token **around a proof net,**
with dynamic rewriting ***guided by the token***

4. *Token-guided* machines

- The token witnesses & controls possible rewriting!

IAM vs. rewrites-first DGoIM

$$(\lambda x.x\,x)\,((\lambda y.y)\,(\lambda z.z)) \;\Downarrow\; \lambda z.z$$



token-passing machine

token-guided machine

slides from CSL 2017 talk
https://www.kurims.kyoto-u.ac.jp/~kmuroya/talks/csl17.pdf

4. *Token-guided* machines

passing a token around a proof net, with dynamic rewriting _guided by the token_

- semantics of functional programs
  - *programs* represented graphically
  - *evaluation* modelled by dynamic computation of the GoI invariant, together with dynamic rewriting of graphs
  - *evaluation strategies* represented by the token

- variants for λ-calculus with different evaluation strategies

  [Sinot, TLCS 2005, with a follow-up paper in MSCS 2006] [M. & Ghica, LMCS 2019]

- application to proving observational equivalence [M., PhD thesis 2020]

4. *Token-guided* machines

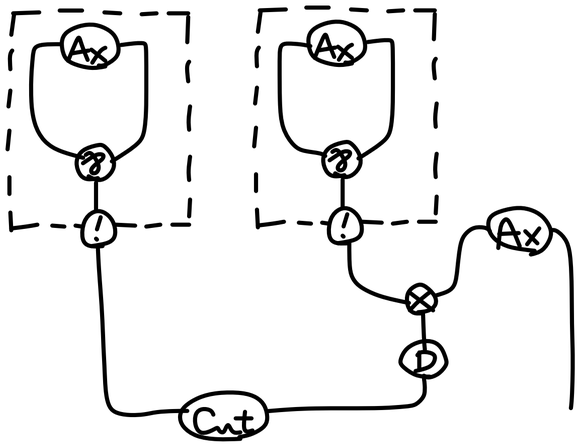
- *evaluation strategies* represented by the token

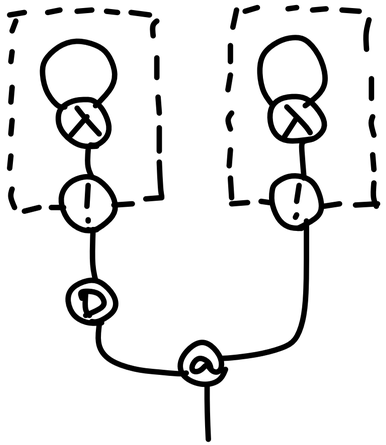
passing a token around a proof net, with dynamic rewriting _guided by the token_

- variants for λ-calculus with different evaluation strategies

  [Sinot, TLCS 2005, with a follow-up paper in MSCS 2006] [M. & Ghica, LMCS 2019]

  - DEMO: https://koko-m.github.io/GoI-Visualiser/





$$(\lambda x\,.\,x)\ (\lambda y\,.\,y)$$

4. *Token-guided* machines

passing a token* around a proof net, with dynamic rewriting *guided by the token

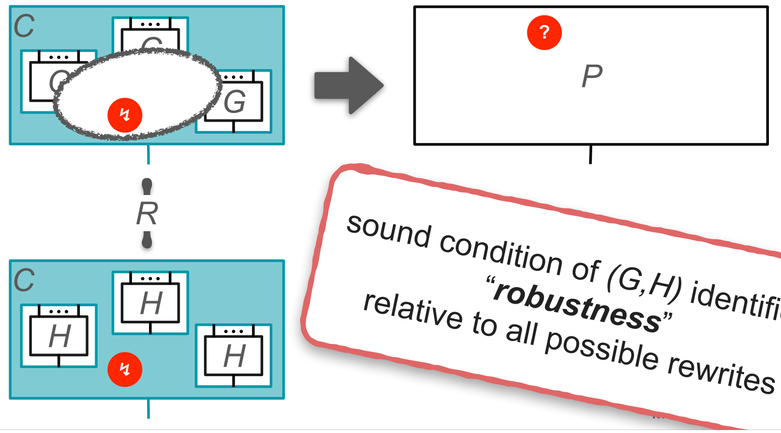
- application to proving observational equivalence [M., PhD thesis 2020]
  - *local* analysis by inspecting the token
    - what happens to a certain part of a program during evaluation
    - how a term interacts with a context

Proof of observational equivalence, using *locality*

Proof idea (simplified):

2. prove that the contextual closure *R* is a **simulation**

Case (3) update of hypernet



sound condition of (G,H) identified: "***robustness***" relative to all possible rewrites

4. *Token-guided* machines

passing a token* around a proof net, with dynamic rewriting *guided by the token

- application to proving observational equivalence [M., PhD thesis 2020]
  - *local* analysis by inspecting the token

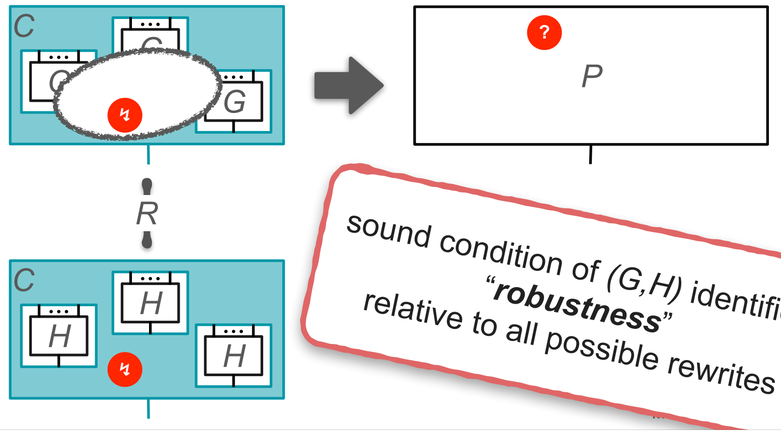
- evaluation strategies represented by the token
- "parts" of a program, represented as sub-graphs (e.g. all parts of a program that refer to the same variable)

Proof of observational equivalence, using *locality*

Proof idea (simplified):

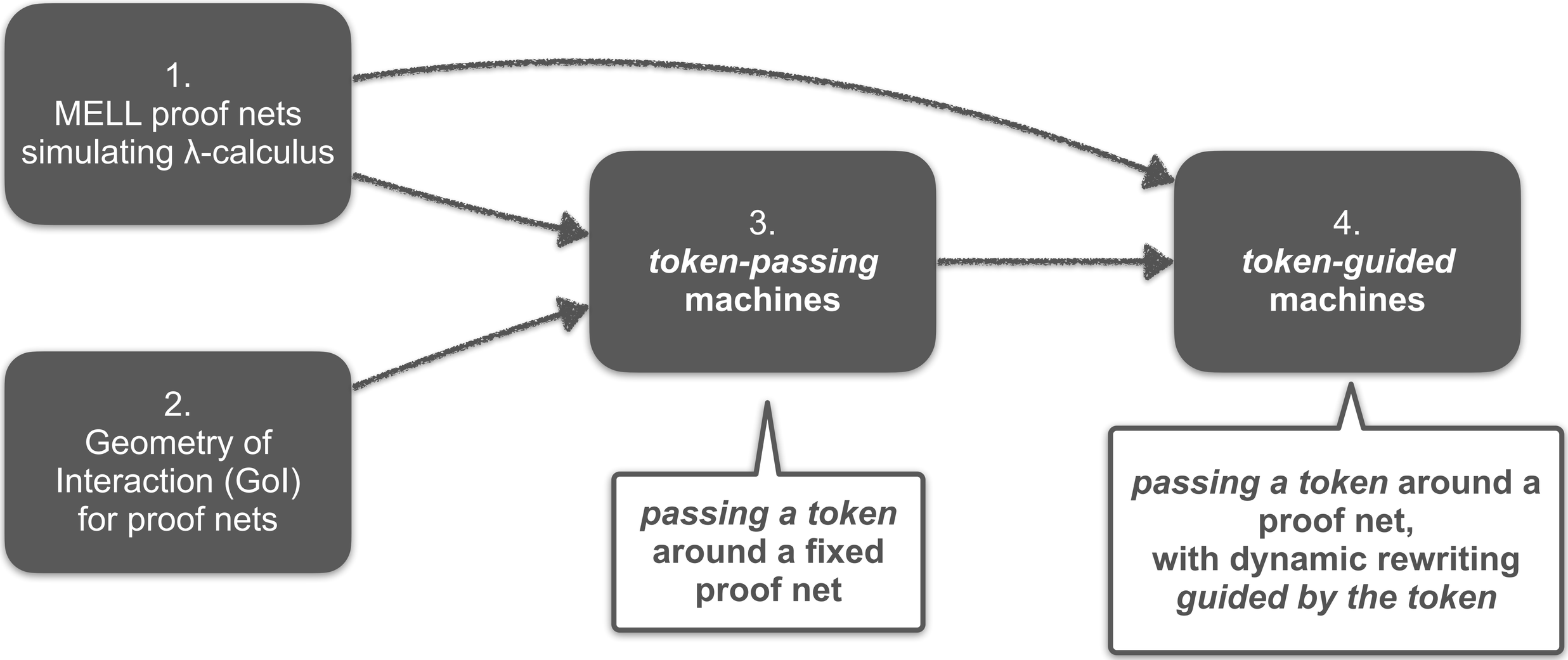
2. prove that the contextual closure *R* is a **simulation**

Case (3) update of hypernet



sound condition of (G,H) identified: "***robustness***" relative to all possible rewrites

Program semantics with token passing: overview



Some research questions

- token-guided machines with *flexible* dynamic rewriting
  - balancing space/time efficiency?

- proving observational equivalence between *effectful* programs with token-guided machines (ongoing)

- transferring insights to conventional syntactical semantics
  - cf. syntactical token-passing machines [Accattoli, Dal Lago & Vanoni, PPDP 2020, with follow-up papers in POPL 2021, LICS 2021]