

Certifying Deadlock Freedom for BIP Models

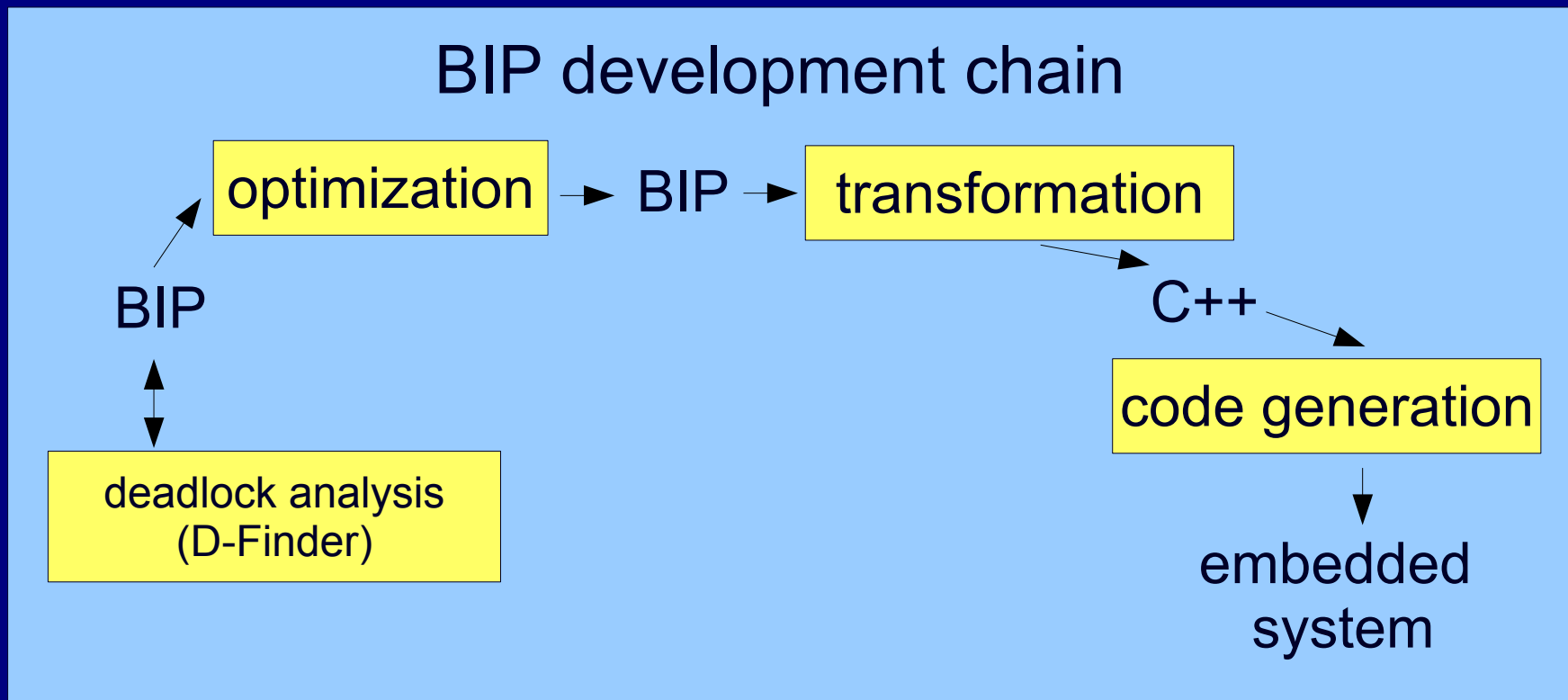
Jan Olaf Blech

Michaël Périn

VERIMAG Laboratory, Université Joseph Fourier, Grenoble

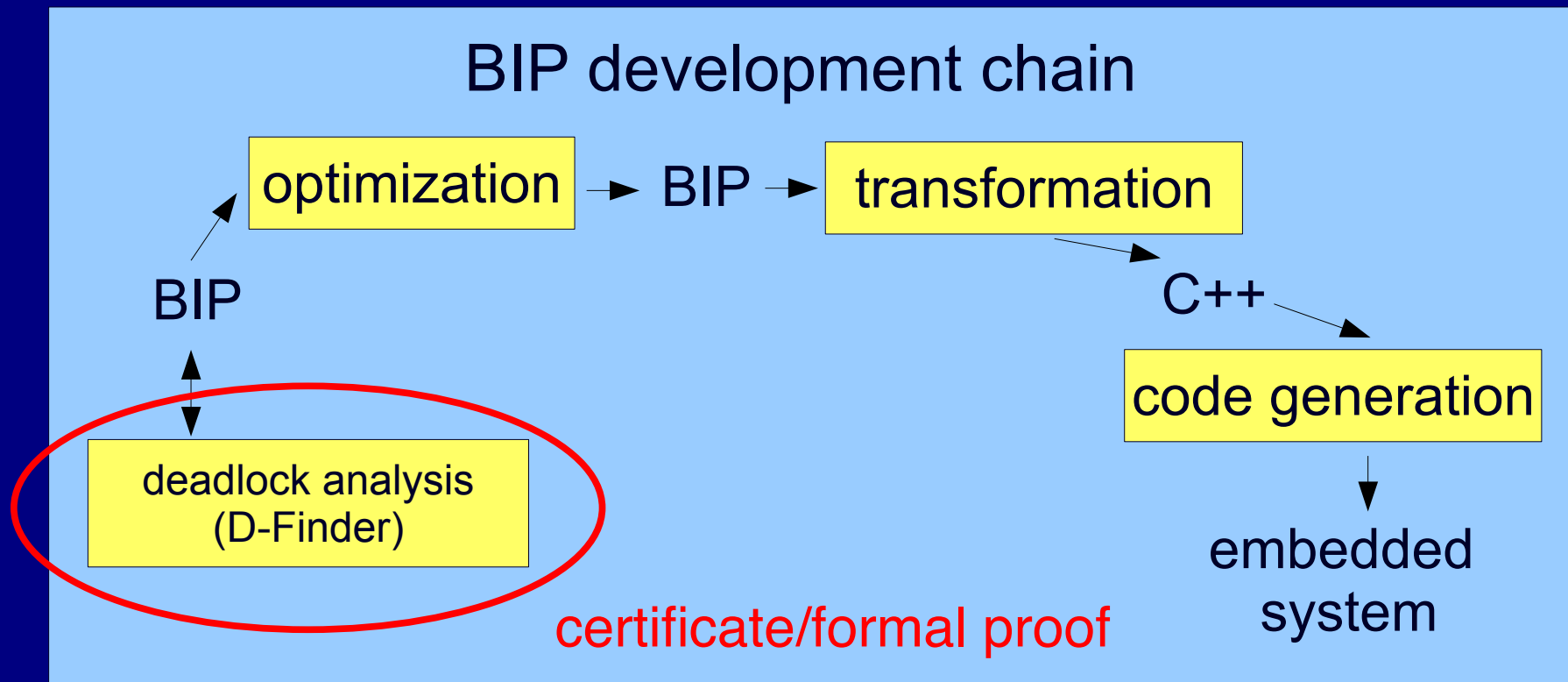
The BIP Tool Chain

- BIP: asynchronous component based language



The BIP Tool Chain

- BIP: asynchronous component based language



Our Approach

- **goal: guarantee / certify deadlock-freedom**



Our Approach

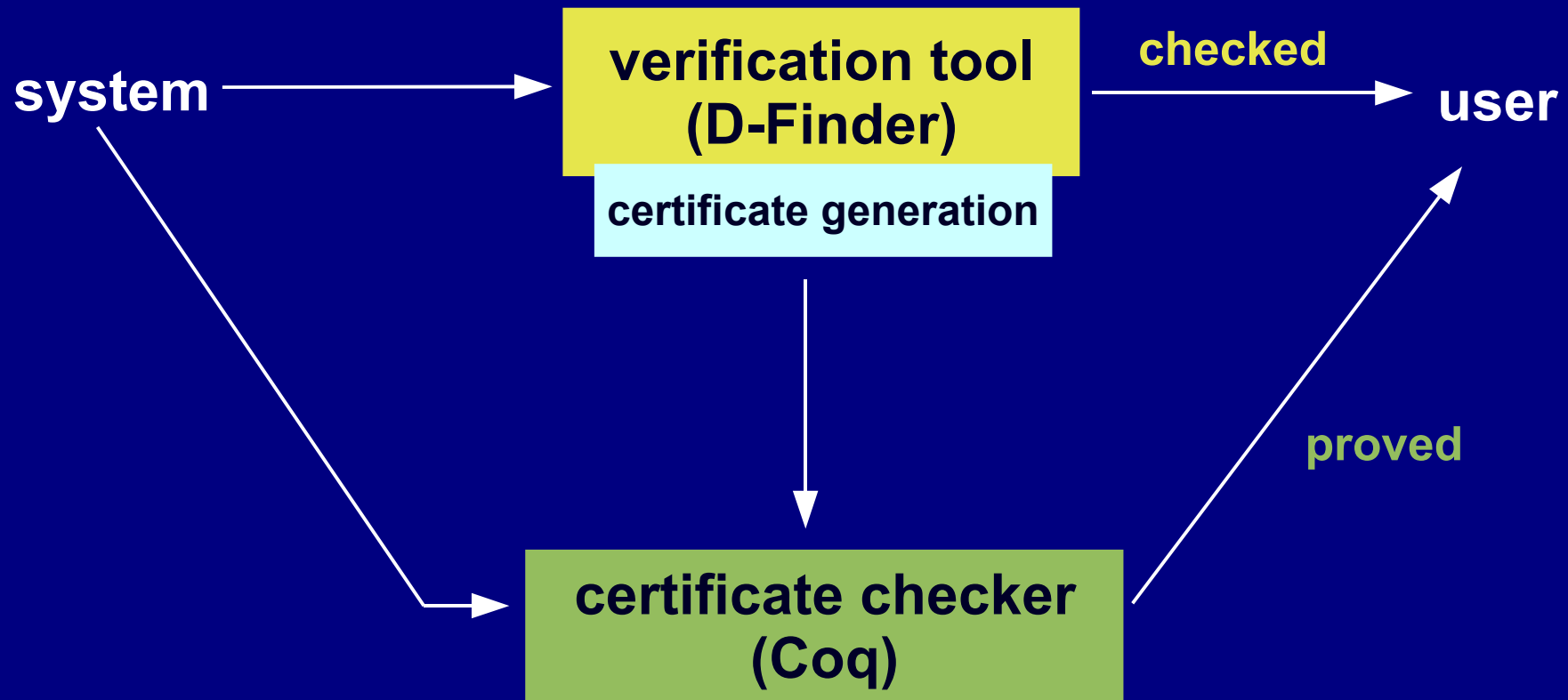
- **goal: guarantee / certify deadlock-freedom**



- **verification tools may contain errors**
 - **wrong results**
 - **not accepted by certification authorities**

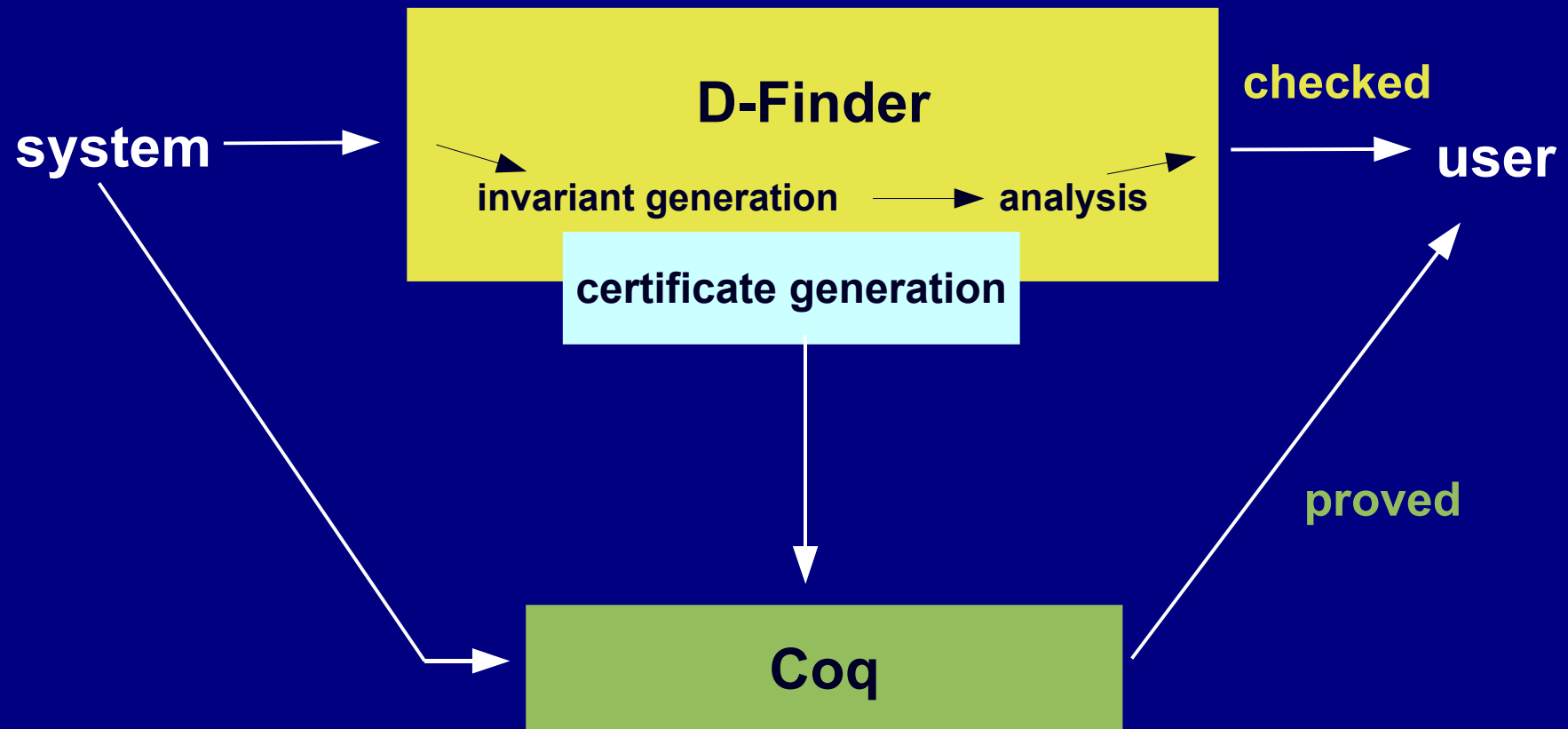
Our Approach

- **verification tools generate certificates**



- **higher level of confidence: Coq is accepted by certification authorities (Common Criteria EAL 7)**

Certificates for D-Finder



Main Characteristics

- certificates are theorem prover proof scripts
 - certificate: property + proof
 - creation by just documenting the discovery process
 - no need to redo tasks that have been done by the verification tool: no need to redo invariant discovery
 - robust to minor implementation changes
 - relatively easy -- “intelligent part” is in the algorithms of the tool
 - general interchange format
 - allows for combination of certificates
 - checking them may be a bottleneck

Main Characteristics

- certificates are theorem prover proof scripts
 - certificate: property + proof main challenge for the verification tool developer
 - creation by just documenting the discovery process
 - no need to redo tasks that have been done by the verification tool: **no need to redo invariant discovery**
 - robust to minor implementation changes
 - relatively easy -- “intelligent part” is in the algorithms of the tool
 - general interchange format
 - allows for combination of certificates
 - checking them may be a bottleneck

main challenge for the certificate infrastructure developer

Certificates for D-Finder

the generated proofs

$$\forall s. \text{Reachable}_{\text{BM}}(s) \rightarrow \text{Enabled}_{\text{BM}}(s)$$



$$\forall s. \text{Reachable}_{\text{BM}}(s) \rightarrow \neg \text{DIS}_{\text{BM}}(s)$$



$$\forall s. \text{Reachable}_{\text{BM}}(s) \rightarrow \text{II}(s) \wedge \text{CI}(s)$$

$$\forall s. \text{II}(s) \wedge \text{CI}(s) \rightarrow \neg \text{DIS}_{\text{BM}}(s)$$



$$\forall s. \text{Reachable}_{\text{BM}}(s) \rightarrow \text{II}(s) \wedge \text{CI}(s)$$

$$\forall s. \neg(\text{II}(s) \wedge \text{CI}(s)) \wedge \text{DIS}_{\text{BM}}(s)$$

Certificates for D-Finder

no deadlocks

$$\forall s. \text{Reachable}_{\text{BM}}(s) \rightarrow \text{Enabled}_{\text{BM}}(s)$$

$$\forall s. \text{Reachable}_{\text{BM}}(s) \rightarrow \neg \text{DIS}_{\text{BM}}(s)$$

$$\forall s. \text{Reachable}_{\text{BM}}(s) \rightarrow \text{II}(s) \wedge \text{CI}(s)$$

$$\forall s. \text{II}(s) \wedge \text{CI}(s) \rightarrow \neg \text{DIS}_{\text{BM}}(s)$$

$$\forall s. \text{Reachable}_{\text{BM}}(s) \rightarrow \text{II}(s) \wedge \text{CI}(s)$$

$$\forall s. \neg(\text{II}(s) \wedge \text{CI}(s)) \wedge \text{DIS}_{\text{BM}}(s)$$

Certificates for D-Finder

$$\forall s. \text{Reachable}_{\text{BM}}(s) \rightarrow \text{Enabled}_{\text{BM}}(s)$$



$$\forall s. \text{Reachable}_{\text{BM}}(s) \rightarrow \neg \text{DIS}_{\text{BM}}(s)$$



$$\forall s. \text{Reachable}_{\text{BM}}(s) \rightarrow \text{II}(s) \wedge \text{CI}(s)$$

$$\forall s. \text{II}(s) \wedge \text{CI}(s) \rightarrow \neg \text{DIS}_{\text{BM}}(s)$$



$$\forall s. \text{Reachable}_{\text{BM}}(s) \rightarrow \text{II}(s) \wedge \text{CI}(s)$$

$$\forall s. \neg(\text{II}(s) \wedge \text{CI}(s) \wedge \text{DIS}_{\text{BM}}(s))$$

most challenging task

Coq Semantics

- operational semantics for flat BIP models
 - atomic components
 - states
 - variables: $(\text{var} \Rightarrow \text{val})$ mapping
 - location
 - transitions
 - source location
 - guard function: $(\text{var} \Rightarrow \text{val}) \Rightarrow \text{bool}$
 - update function: $(\text{var} \Rightarrow \text{val}) \Rightarrow (\text{var} \Rightarrow \text{val})$
 - port
 - target location
 - composed components
 - states: list of atomic components' states
 - interactions: list of ports
 - semantics: 1 inference rule

Coq Semantics

- operational semantics for flat BIP models

- atomic components

- states

- variables: $(\text{var} \Rightarrow \text{val})$ mapping
 - location

- transitions

- source location
 - guard function: $(\text{var} \Rightarrow \text{val}) \Rightarrow \text{bool}$
 - update function: $(\text{var} \Rightarrow \text{val}) \Rightarrow (\text{var} \Rightarrow \text{val})$
 - port
 - target location

- composed components

- states: list of atomic components' states
 - interactions: list of ports
 - semantics: 1 inference rule

state transition
systems

combination

Proving an Inductive Invariant

- verification goal

$\forall s. \text{Reachable}_{\text{BM}}(s) \rightarrow$

$\text{CI}_1(s) \wedge \dots \wedge \text{CI}_n(s) \wedge \text{II}_1(s) \wedge \dots \wedge \text{II}_m(s)$

conjunction of invariants

$a_1(s) \vee \dots \vee a_j(s)$

prove invariants independently

Proving an Inductive Invariant

- induction

$$\Rightarrow (a_1(\text{init}) \vee \dots \vee a_n(\text{init}))$$

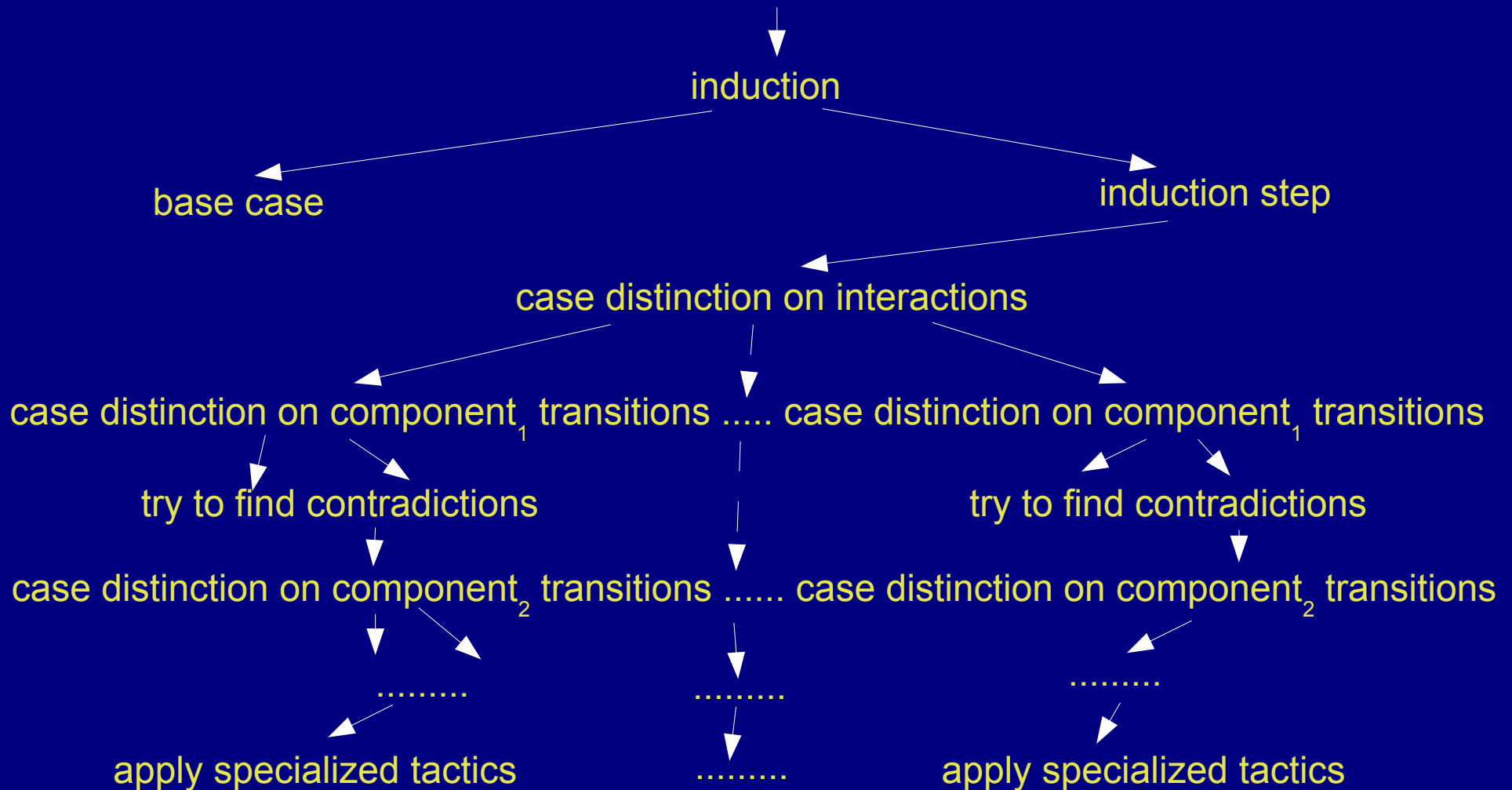
$$\Rightarrow (a_1(s) \vee \dots \vee a_n(s))$$

$$s \rightarrow s'$$

$$(a_1(s') \vee \dots \vee a_n(s'))$$

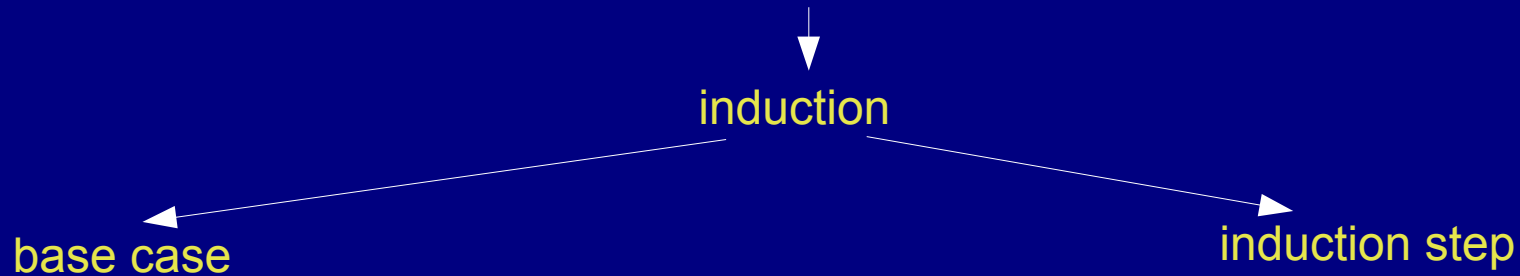
Proof Script Generation

$$\forall s. \text{Reachable}_{\text{BM}}(s) \rightarrow \text{II}(s) \wedge \text{CI}(s)$$



Proof Script Generation

$$\forall s. \text{Reachable}_{\text{BM}}(s) \rightarrow \text{II}(s) \wedge \text{CI}(s)$$



case distinction on interactions

mostly based on BIP model

case distinction on component₁ transitions case distinction on component₁ transitions

try to find contradictions

try to find contradictions

case distinction on component₂ transitions case distinction on component₂ transitions

apply specialized tactics

apply specialized tactics

Evaluation

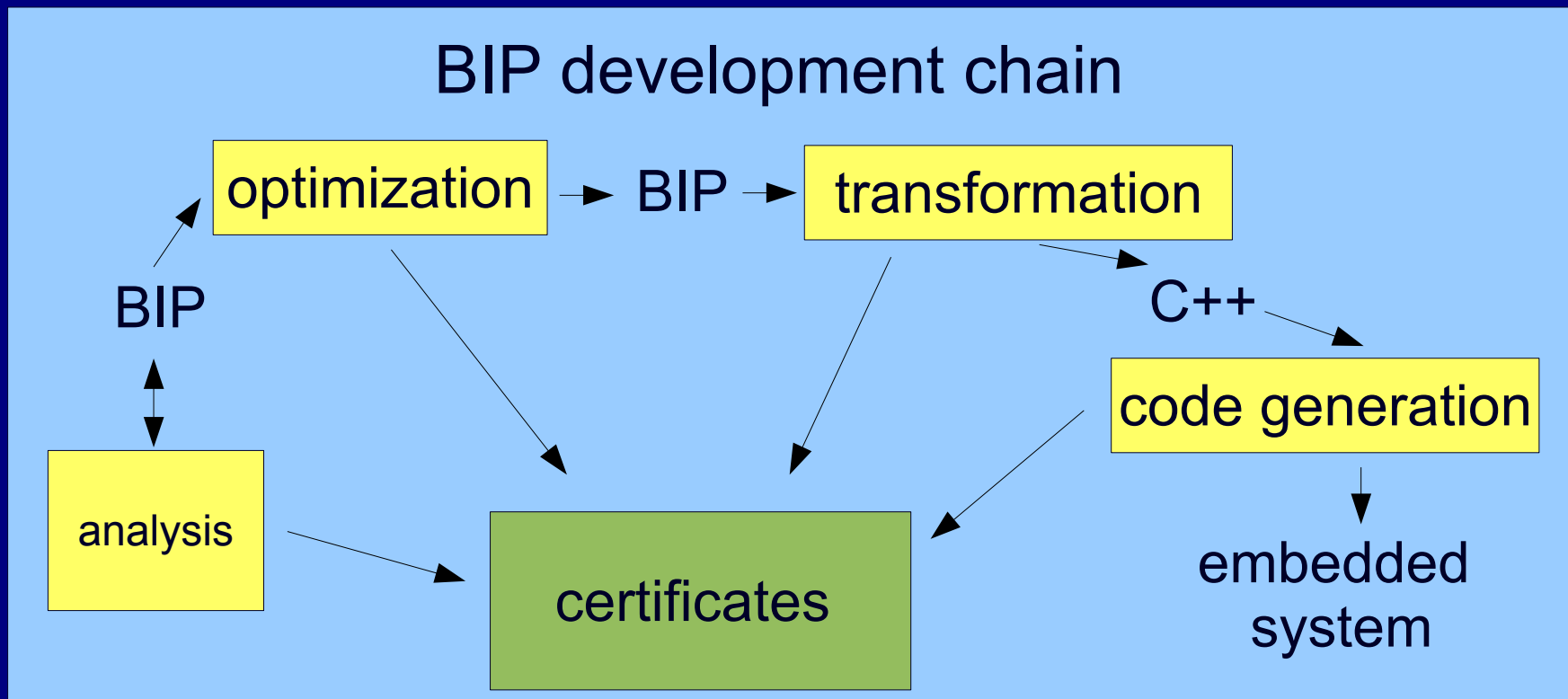
- what has been implemented
 - (subset of) BIP semantics for Coq
 - Coq representation generation implemented in Java for (a subset of) BIP based on Java library for BIP2
 - automatic proof script generation for invariants based on invariants provided by D-Finder
 - implemented in Ocaml
 - needs some manual instantiation for certain guard and update expressions
- a few minutes checking time for small BIP models

Related Approaches / Work

- (Foundational) Proof Carrying Code
[Necula, Appel,...]
- Translation Validation
 - classical approach [Pnueli, Zuck, ...]
 - scheduling algorithm in Compcert [Tristan + Leroy '08]
- documenting results of verification tools
 - model checkers [Namjoshi, Cleaveland...]
 - SAT solver [Zhang + Malik '03]

Vision: A Certificate Generating BIP Tool Chain

- certifying analysis results and transformations



- lift correctness results through the development chain

Future Work (cont.)

- BIP semantics
 - hierarchical components
 - exploit semantic features in certificate checking
 - higher programming language guards updates + methods to reason about them explicitly
- dealing with non-inductive BIP invariants
- complete deadlock certificate infrastructure
- improve checking speed

Thank you for your attention!