# Abstraction Refinement for Hybrid Systems

André Platzer

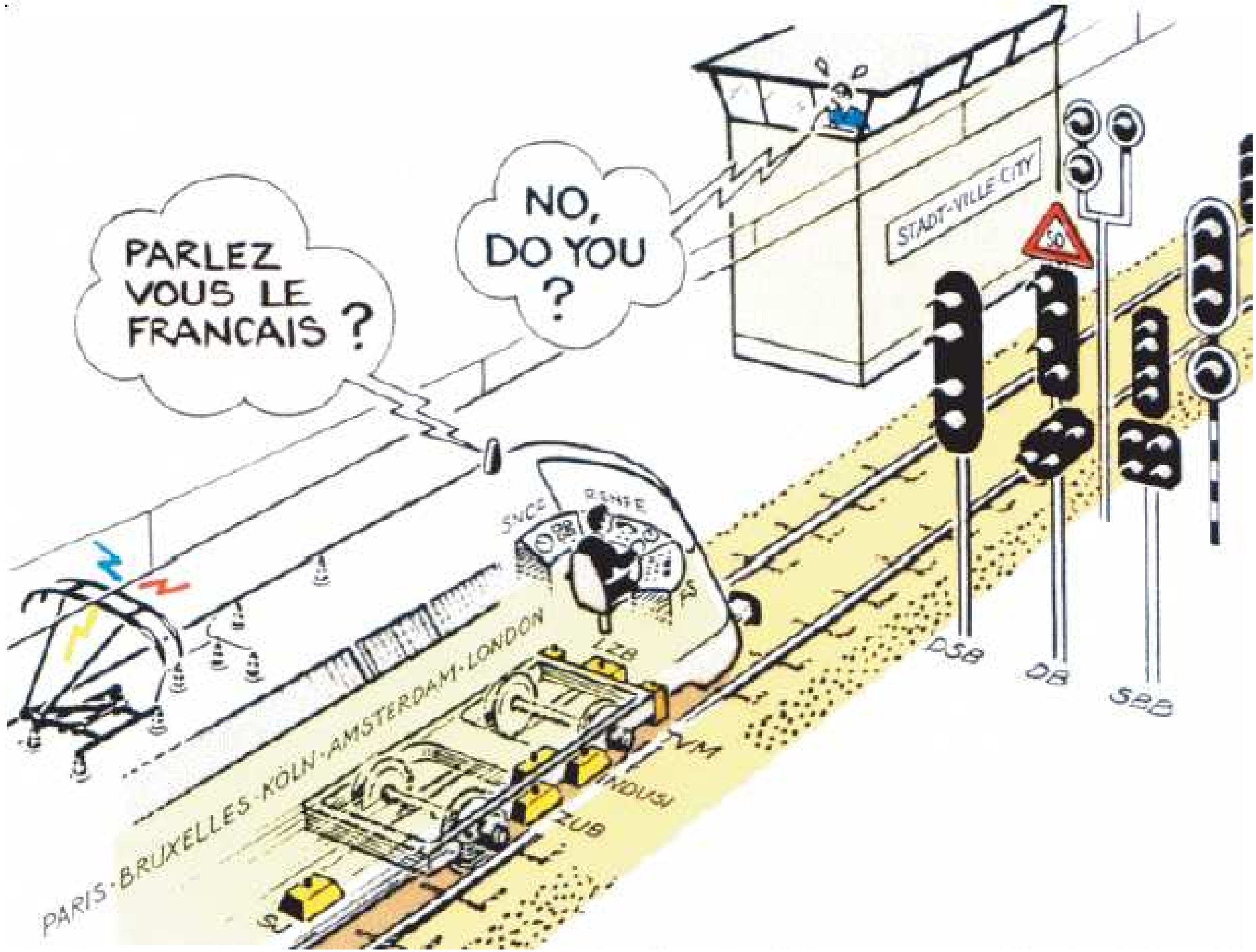
University of Oldenburg

# SFB AVACS

Automatic Verification and Analysis of Complex Systems

| Project Areas: | Case Study: |
| --- | --- |
| **R**eal-Time Systems | European |
| **H**ybrid Systems | Train Control System |
| **S**ystem Construction | (ETCS) |

# EU Train Control System



# Overview

- Abstraction Refinement
- Concept
- Craig Interpolation
- Hybrid Systems
- Hybrid Dynamic Logic

# Abstraction Refinement

Part I: Abstraction Refinement

# Abstraction

concrete

$\mathcal{K} \models \neg(\exists\Diamond B)$

# Abstraction

| concrete | abstract |
|---|---|
| $\mathcal{K} \models \neg(\exists\Diamond B)$ | $\tilde{\mathcal{K}} \models \neg(\exists\Diamond B)$ |

where abstract $\tilde{\mathcal{K}}$ is simpler than model $\mathcal{K}$

# Abstraction

| concrete | abstract |
| --- | --- |

$\mathcal{K} \models \neg(\exists\Diamond B) \quad \Leftarrow \quad \tilde{\mathcal{K}} \models \neg(\exists\Diamond B) \qquad \surd$

where abstract $\tilde{\mathcal{K}}$ is simpler than model $\mathcal{K}$

# Abstraction

| concrete | | abstract | |
|---|---|---|---|
| $\mathcal{K} \models \neg(\exists\Diamond B)$ | $\Leftarrow$ | $\tilde{\mathcal{K}} \models \neg(\exists\Diamond B)$ | $\surd$ |
| $\mathcal{K} \not\models \neg(\exists\Diamond B)$ | | | |

where abstract $\tilde{\mathcal{K}}$ is simpler than model $\mathcal{K}$

# Abstraction

| concrete | | abstract | |
|---|---|---|---|
| $\mathcal{K} \models \neg(\exists\Diamond B)$ | $\Leftarrow$ | $\tilde{\mathcal{K}} \models \neg(\exists\Diamond B)$ | $\surd$ |
| $\mathcal{K} \not\models \neg(\exists\Diamond B)$ | $\Rightarrow$ | $\tilde{\mathcal{K}} \not\models \neg(\exists\Diamond B)$ | $\times$ |

where abstract $\tilde{\mathcal{K}}$ is simpler than model $\mathcal{K}$

# Abstraction

| concrete | | abstract | |
|---|---|---|---|
| $\mathcal{K} \models \neg(\exists\Diamond B)$ | $\Leftarrow$ | $\tilde{\mathcal{K}} \models \neg(\exists\Diamond B)$ | $\surd$ |
| $\mathcal{K} \not\models \neg(\exists\Diamond B)$ | $\Rightarrow$ | $\tilde{\mathcal{K}} \not\models \neg(\exists\Diamond B)$ | $\times$ |
| $?$ | $\Leftarrow$ | $\tilde{\mathcal{K}} \not\models \neg(\exists\Diamond B)$ | |

where abstract $\tilde{\mathcal{K}}$ is simpler than model $\mathcal{K}$

# Abstraction Refinement

| concrete | | abstract | |
|---|---|---|---|
| $\mathcal{K} \models \neg(\exists\Diamond B)$ | $\Leftarrow$ | $\tilde{\mathcal{K}} \models \neg(\exists\Diamond B)$ | $\surd$ |
| $\mathcal{K} \not\models \neg(\exists\Diamond B)$ | $\Rightarrow$ | $\tilde{\mathcal{K}} \not\models \neg(\exists\Diamond B)$ | $\times$ |
| $?$ | $\Leftarrow$ | $\tilde{\mathcal{K}} \not\models \neg(\exists\Diamond B)$ | by trace $\tilde{t}$ |

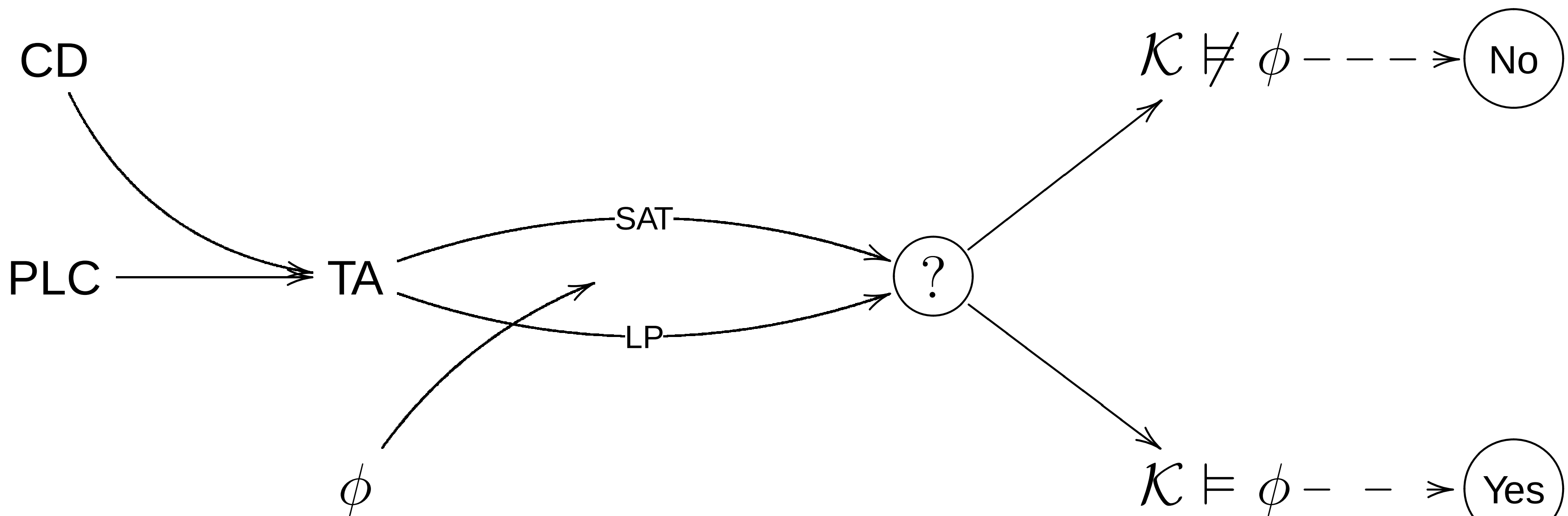
where abstract $\tilde{\mathcal{K}}$ is simpler than model $\mathcal{K}$

Refine if $\tilde{\mathcal{K}}, \tilde{t} \not\models \neg(\exists\Diamond B)$ but $\mathcal{K}, \tilde{t} \models \neg(\exists\Diamond B)$

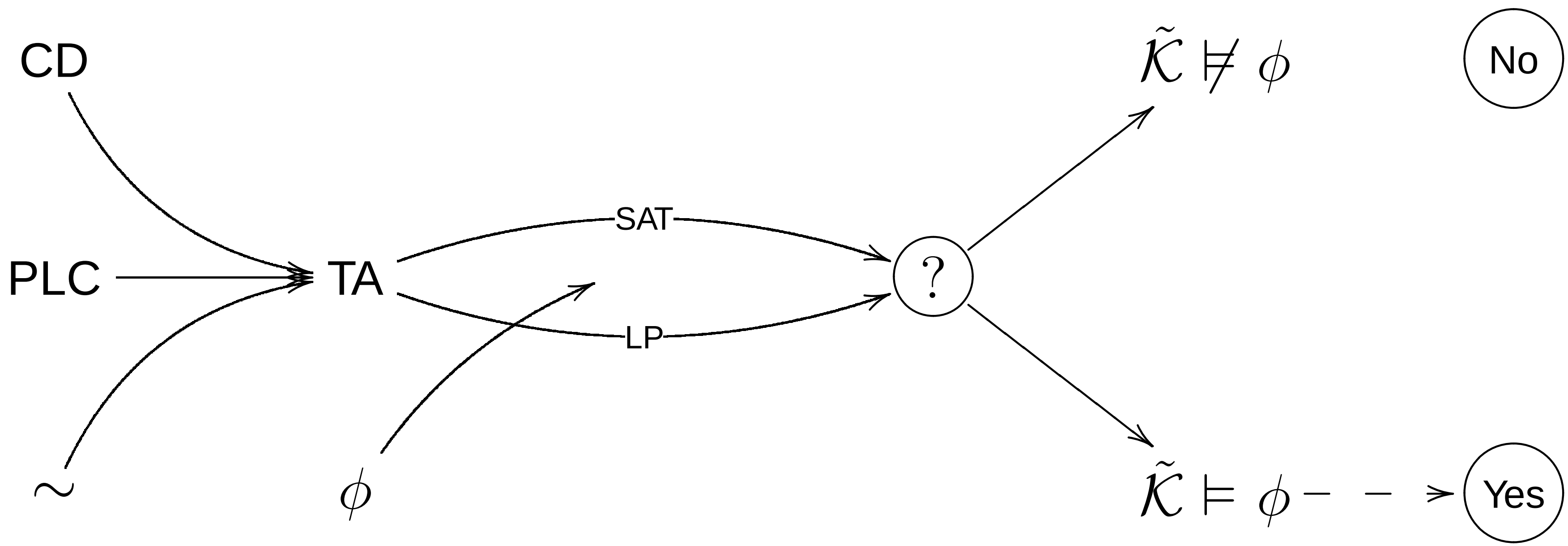
# Guiding Principle

- "Abstraction is deductive reasoning" (Podelski *et al*).
- Reason why abstract trace infeasible encoded in infeasibility proof of the trace (Henzinger *et al.*, 2004).
  $\rightsquigarrow$ Craig interpolation for refinement.
- BMC quick for falsification.

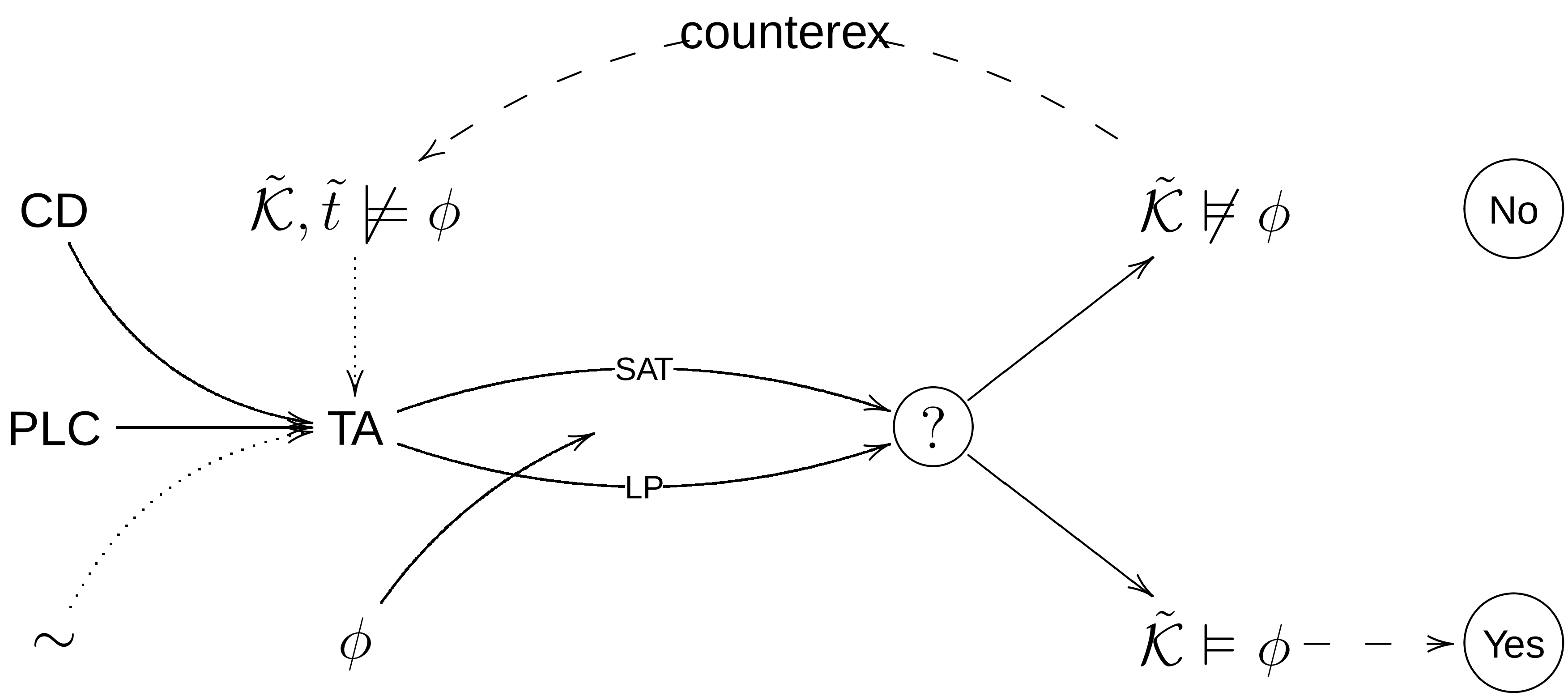
# Architecture



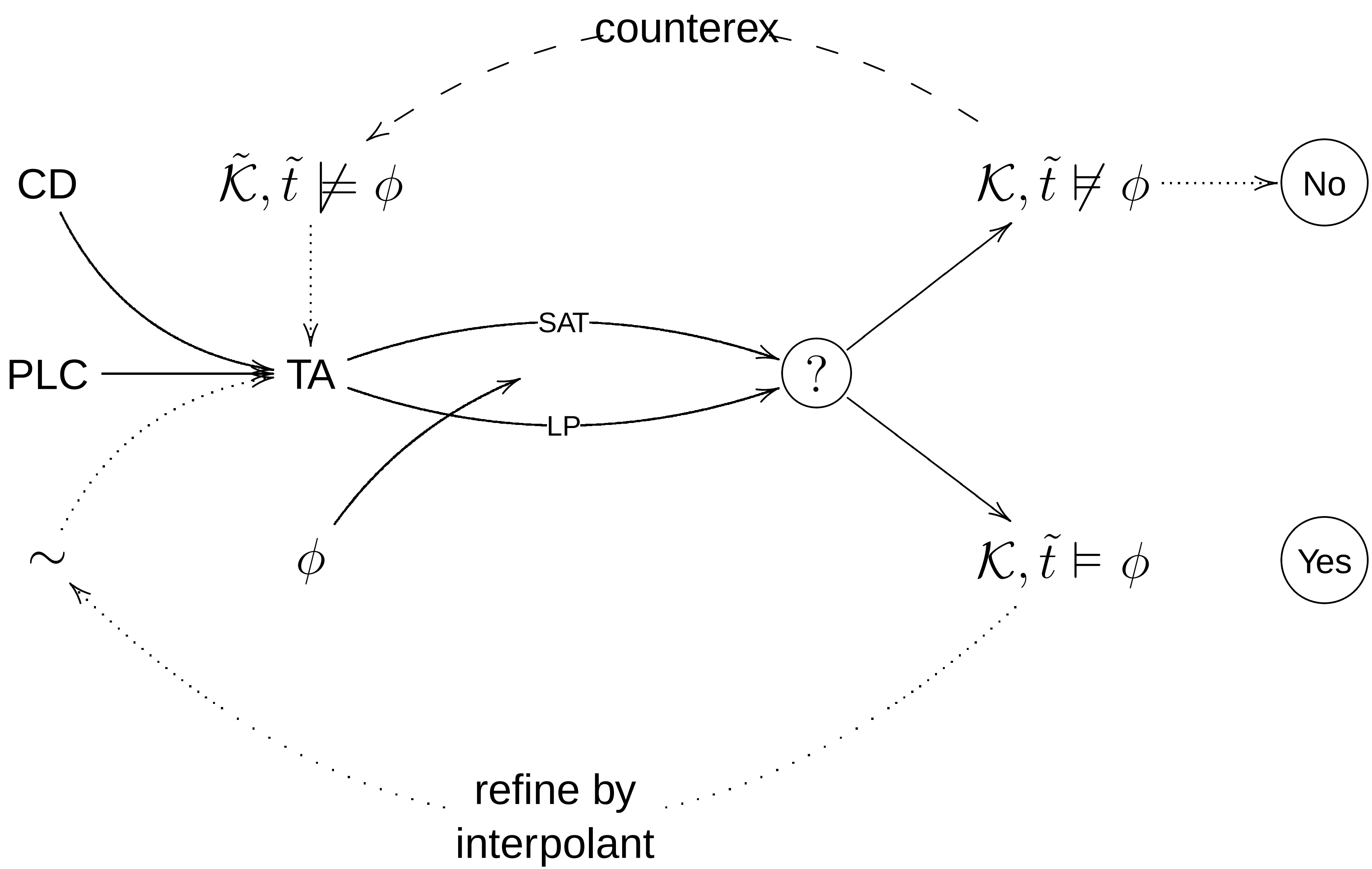
# Abstr. Architecture



# Abstr. Architecture

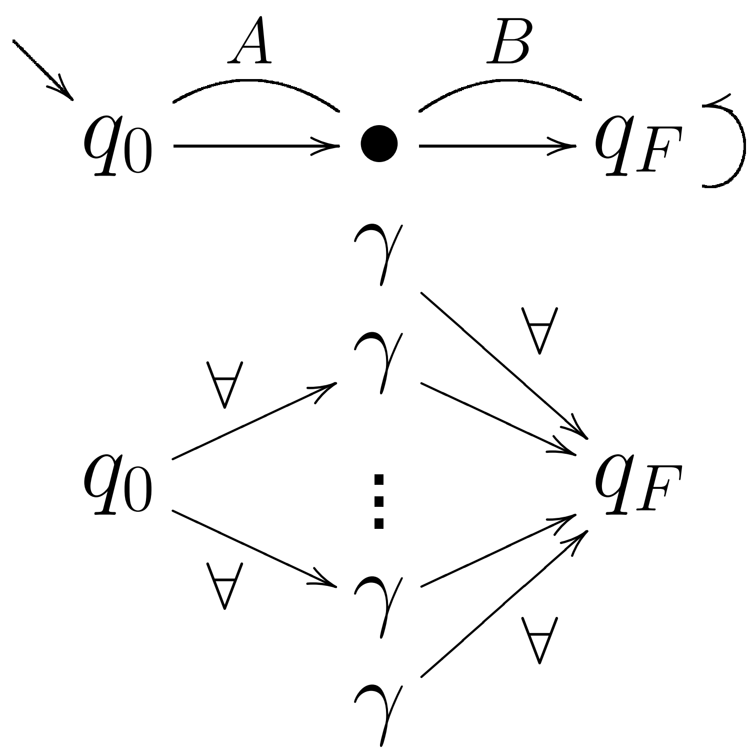


# Abstr. Refin. Architecture



# Craig Interpolation

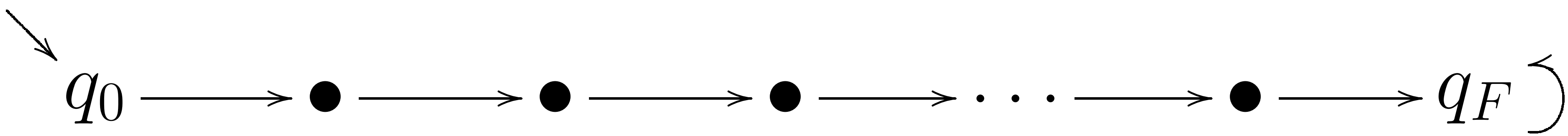
- $A \models B$ then $\exists\gamma$ of common vocabulary s.t. $A \models \gamma$ and $\gamma \models B$.




$\gamma$ characteristic connection

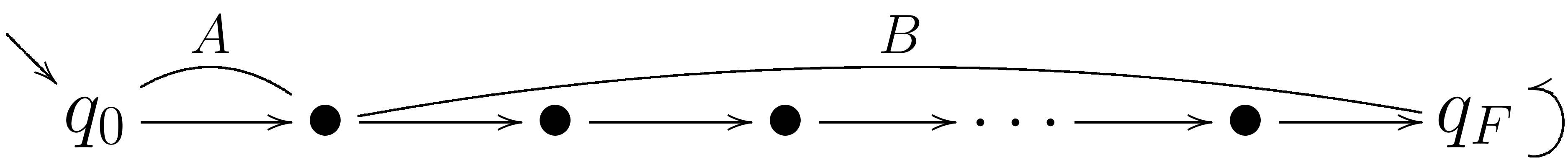
# Craig Interpolation

- $A \models B$ then $\exists \gamma$ of common vocabulary s.t. $A \models \gamma$ and $\gamma \models B$.



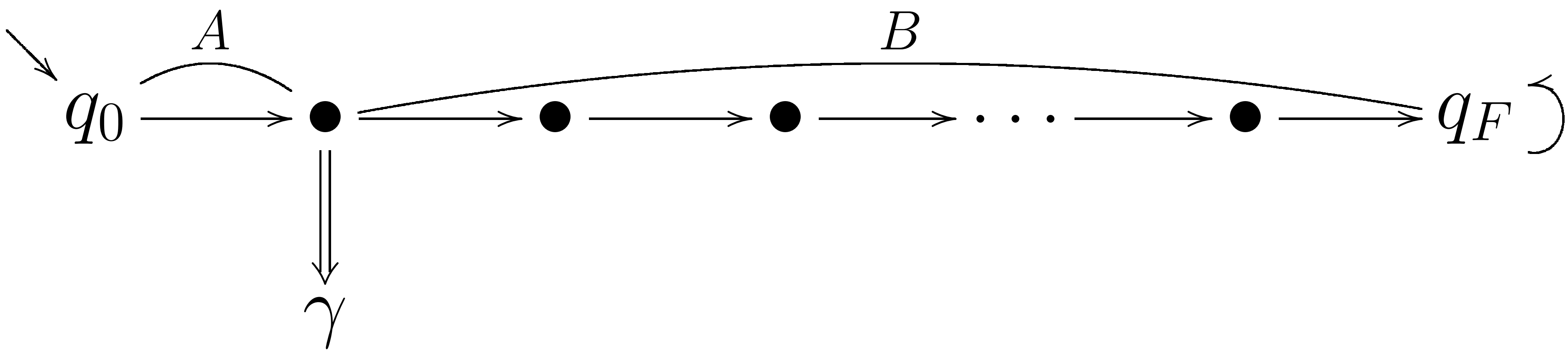
# Craig Interpolation

- $A \models B$ then $\exists \gamma$ of common vocabulary s.t. $A \models \gamma$ and $\gamma \models B$.



# Craig Interpolation

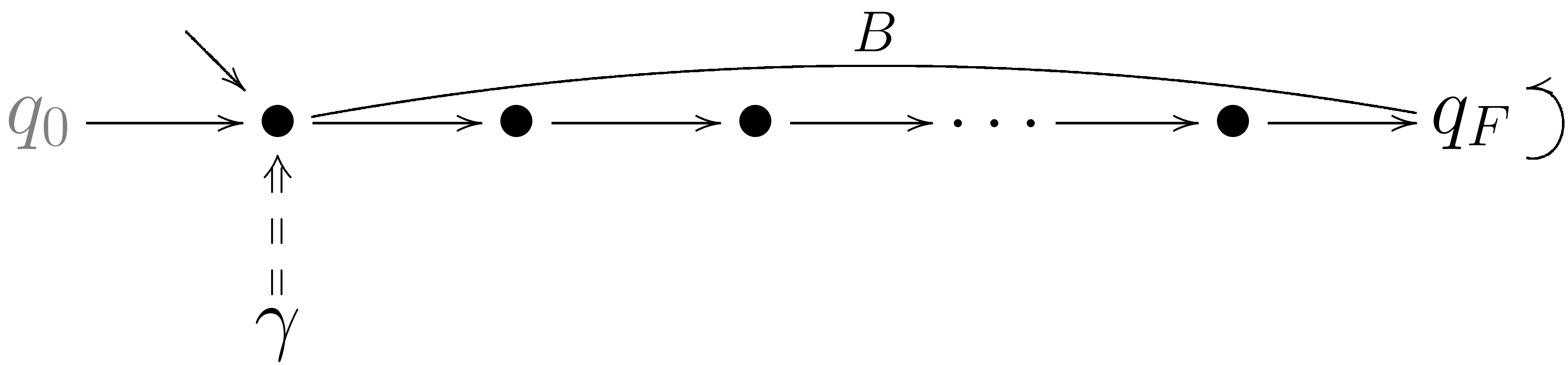
- $A \vDash B$ then $\exists \gamma$ of common vocabulary s.t. $A \vDash \gamma$ and $\gamma \vDash B$.




- $\gamma$ characteristic for $\mathcal{T}(q_0)$ in $\mathcal{T}$.

# Craig Interpolation

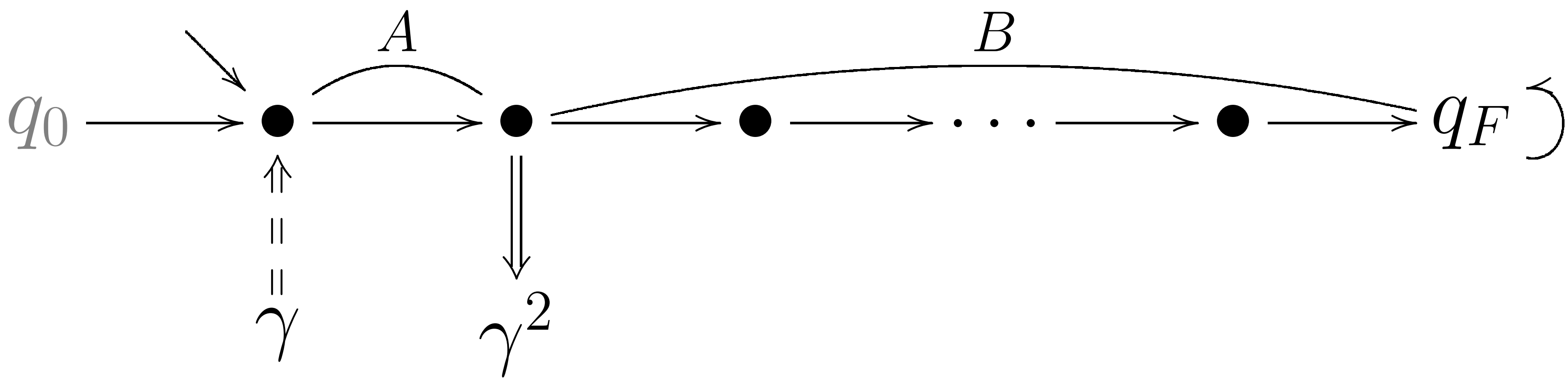
- $A \models B$ then $\exists\gamma$ of common vocabulary s.t. $A \models \gamma$ and $\gamma \models B$.




- $\gamma$ characteristic for $\mathcal{T}(q_0)$ in $\mathcal{T}$.

# Craig Interpolation

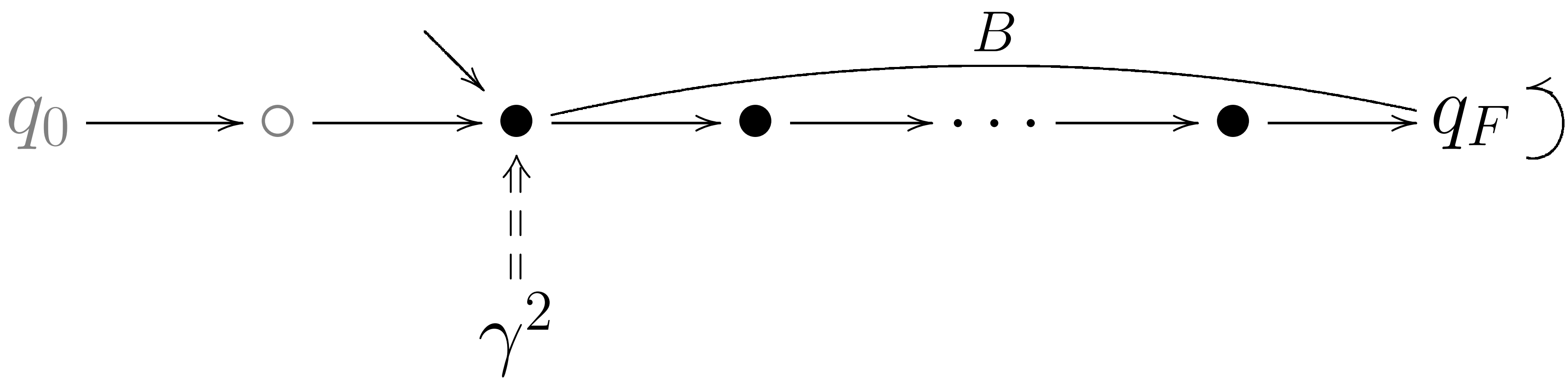
- $A \vDash B$ then $\exists \gamma$ of common vocabulary s.t. $A \vDash \gamma$ and $\gamma \vDash B$.




- $\gamma^2$ characteristic for $\mathcal{T}^2(q_0)$ in $\mathcal{T}$.

# Craig Interpolation

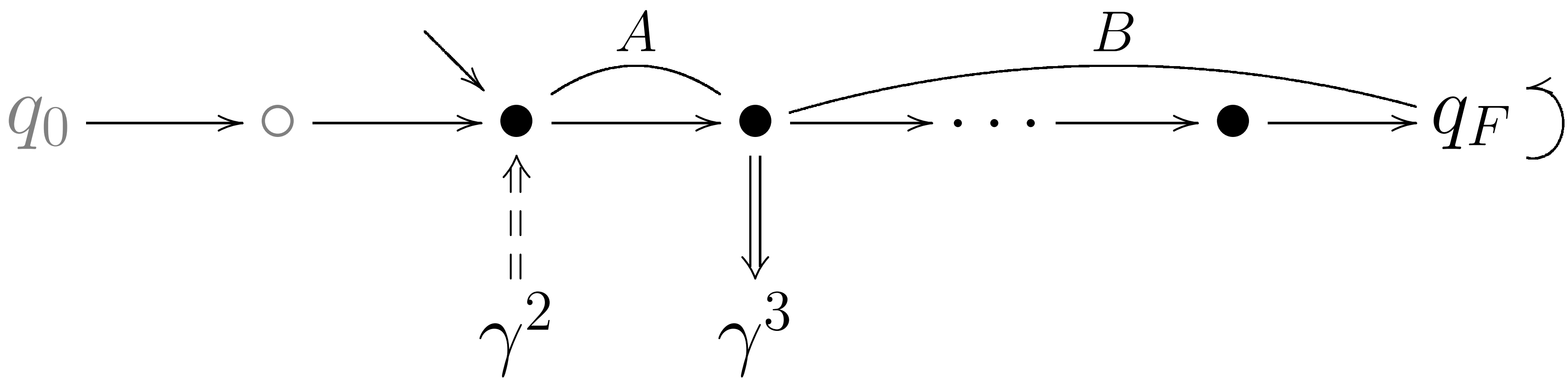
- $A \models B$ then $\exists\gamma$ of common vocabulary s.t. $A \models \gamma$ and $\gamma \models B$.




- $\gamma^2$ characteristic for $\mathcal{T}^2(q_0)$ in $\mathcal{T}$.

# Craig Interpolation

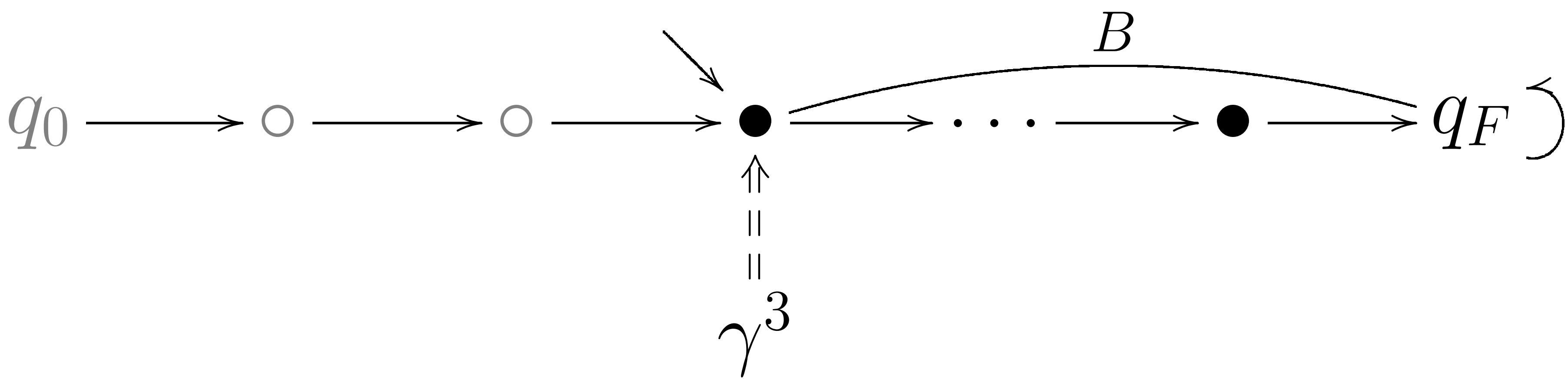
- $A \models B$ then $\exists\gamma$ of common vocabulary s.t. $A \models \gamma$ and $\gamma \models B$.




- $\gamma^k$ characteristic for $\mathcal{T}^k(q_0)$ in $\mathcal{T}$.

# Craig Interpolation

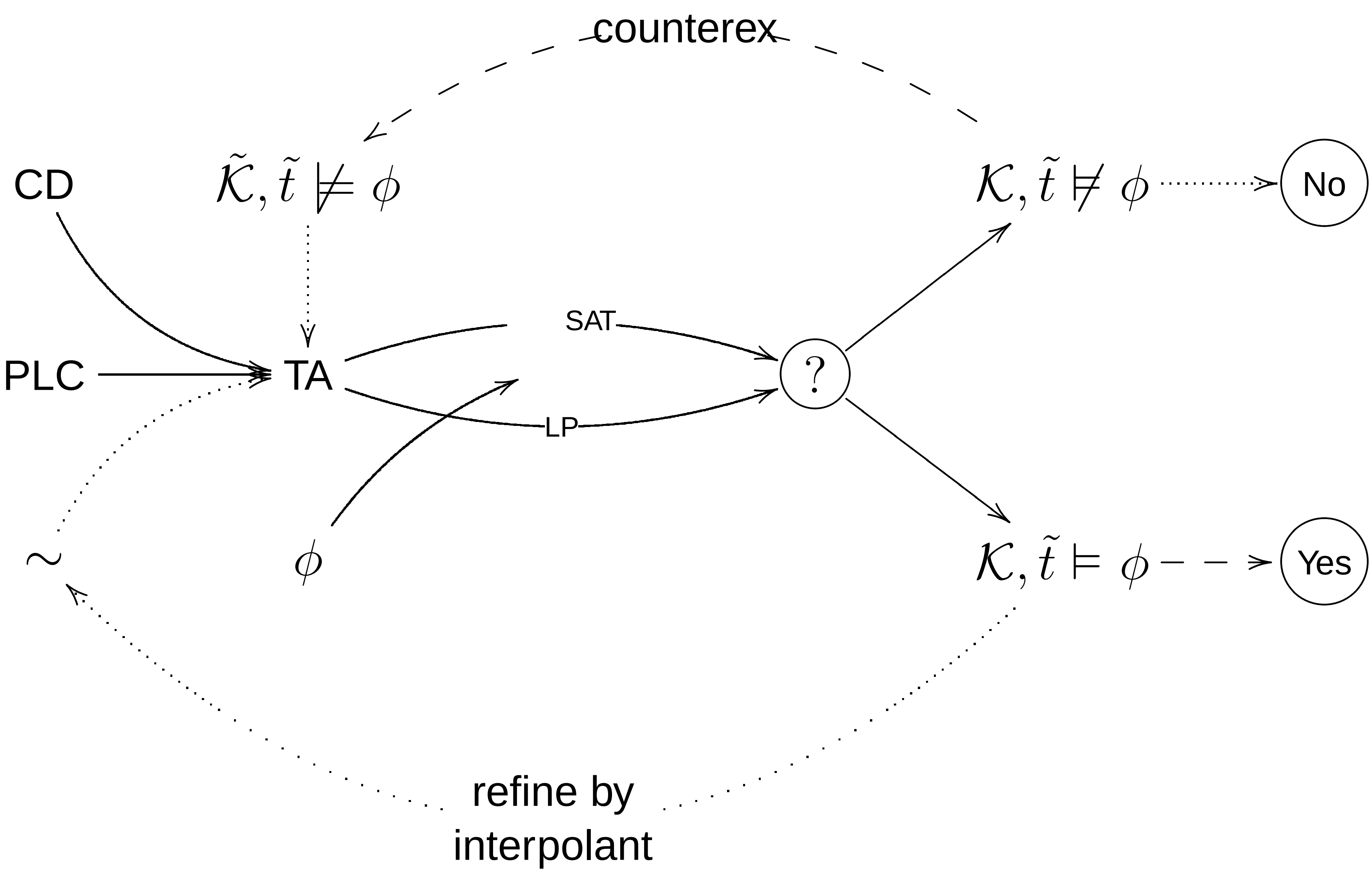
- $A \models B$ then $\exists \gamma$ of common vocabulary s.t. $A \models \gamma$ and $\gamma \models B$.



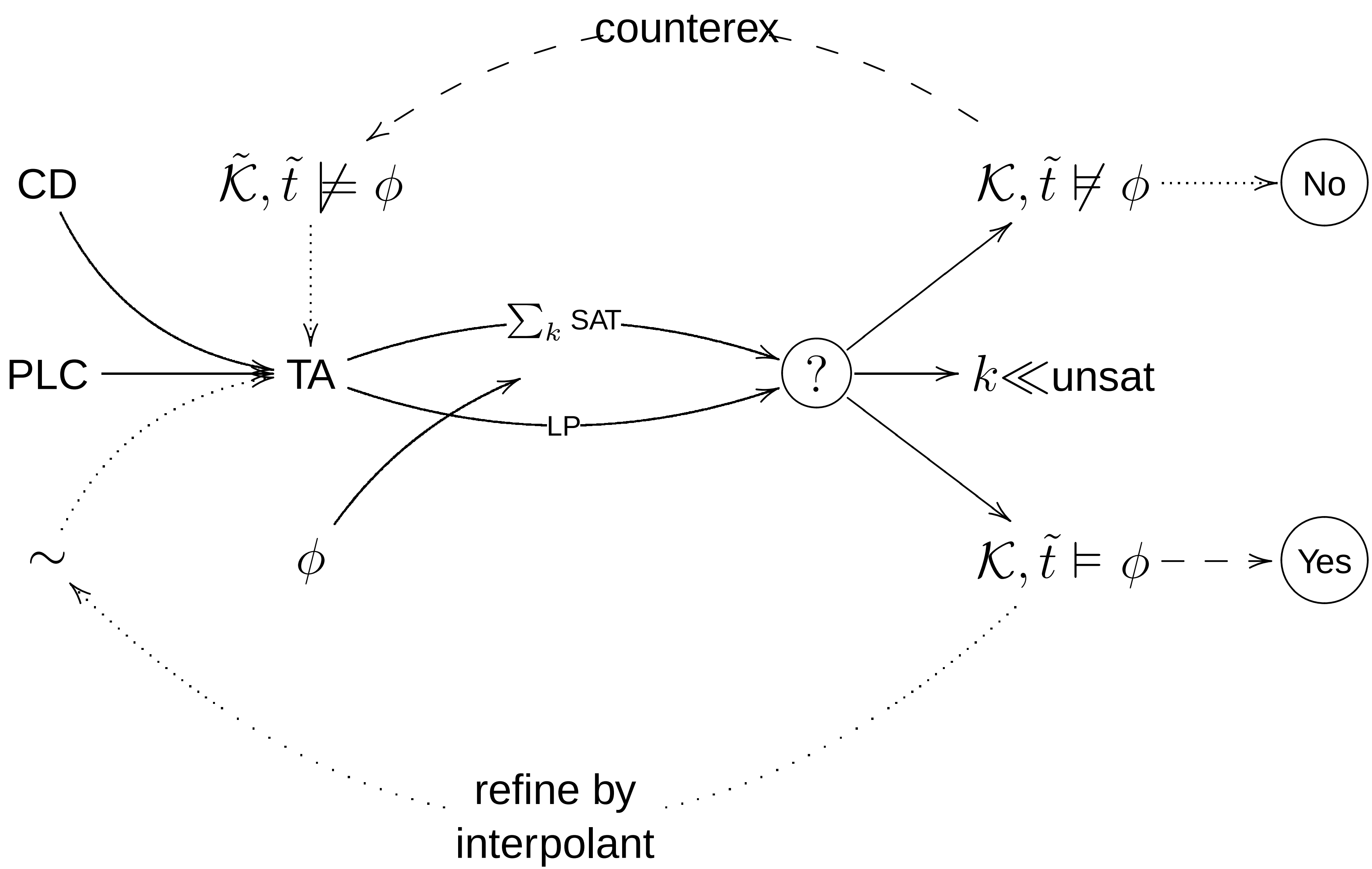

- $\gamma^k$ characteristic for $\mathcal{T}^k(q_0)$ in $\mathcal{T}$.

$\Rightarrow$ $\gamma^{dia}$ characterises reachable final states in $\mathcal{T}$. Characterises concretisability.

# Abstr. Refin. Architecture



# Abstr. Refin. Architecture



# Features of Architecture

- MC with adv. Constraint+SAT-solver.
- Counterexample loop concretises traces.
- Generalising refinement loop for spurious traces by interpolation.
- Interpolation with enhanced SAT-prover techniques.
- Bounded model checking necessary for SAT+LP encoding.

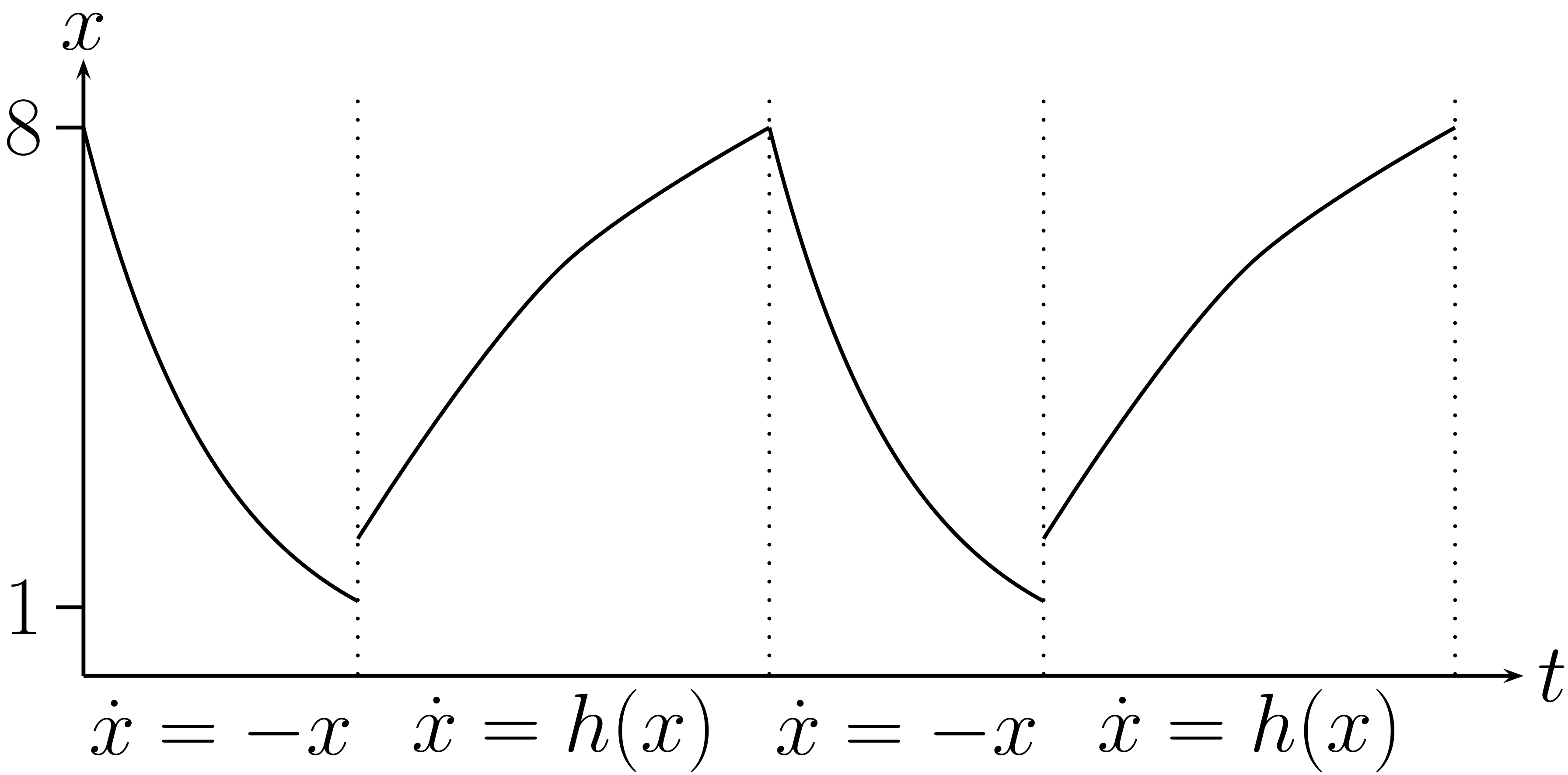
# Extensions

- 2LP: two-variable case $x \leq y + 2$ rather than full LP.
- Bound distinction by interpolation?
- Interpolating Model Checking (McMillan, 2003).
- Minimal interpolants for generalisable concretion?

# Hybrid Systems

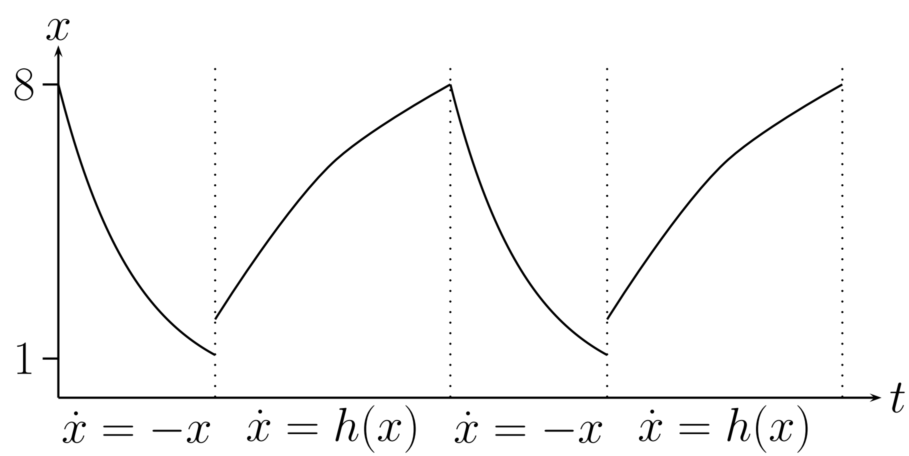
Part II: Hybrid Systems

# Hybrid System Components



# Hybrid Dynamic Logic

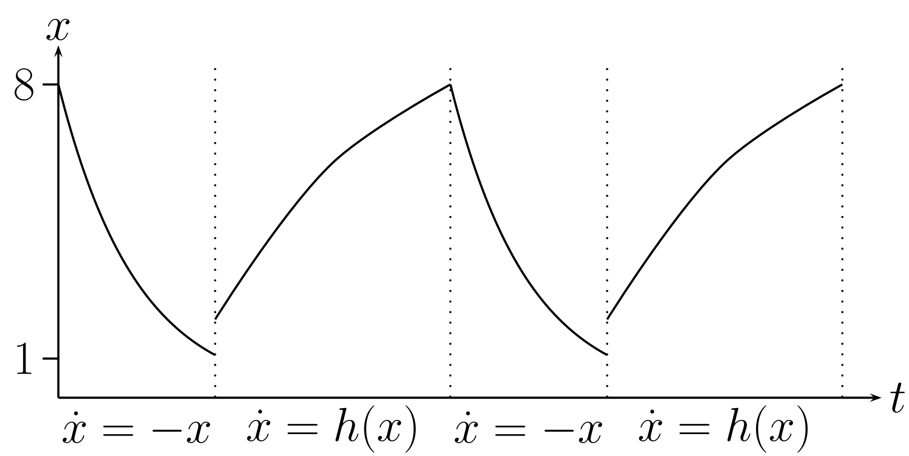
- $x \leq 8 \;\rightarrow\; [\dot{x} = -x]x \leq 8$
- Invariant evolution:
  $x = x_0 \;\rightarrow\; [\dot{x} = -x \,\&\, x \geq 1]x \leq x_0$




$$[\;\dot{x} = -x \,\&\, x{\geq}1;\; x{\leq}2?;\; x := x{+}1;\; \dot{x} = h(x)\;\;]\phi$$

# Hybrid Dynamic Logic

- $x \leq 8 \;\rightarrow\; [\dot{x} = -x]x \leq 8$
- Invariant evolution:
  $x = x_0 \;\rightarrow\; [\dot{x} = -x \,\&\, x \geq 1]x \leq x_0$




$$[(\dot{x} = -x \,\&\, x{\geq}1;\; x{\leq}2?;\; x := x{+}1;\; \dot{x} = h(x))^*]\phi$$

# Atomic Reduction

- $x \leq 8 \rightarrow [\dot{x} = -x]x \leq 8$

$\rightsquigarrow x \leq 8 \rightarrow \forall t \, (x' = c \cdot e^{-t} \rightarrow x' \leq 8)$

HyDyn

$\downarrow$ DGT

FOL$[\mathbf{R}, +, \cdot, e]$

# Summary (I)

- Abstraction refinement by Craig interpolation.
- Systematic concept.
- Refinement problem is effective.
- Interpolant *characteristic* connection.

? Flexible bounds?

? Interpolant feature extraction?

# Summary (II)

- Hybrid verification involves physical models.
- Isolate verification components.
- Frugal language expressing hybrid properties $\rightsquigarrow$ dynamic logic.

? Verify/prove with
  - solution via differential Galois theory
  - “flow characteristics” of the DES itself
  - Abstraction refinement

# Discussion

Discussion

# Repository

The end of the presentation

# Dimension

- Complexity of $k$ unfolding

| | |
|---|---|
| formula | $O(kT^2)$ |
| $\mathbb{R}$-variables | $O(kX)$ |
| prop. variables | $O(k(\log S + \log A + T))$ |

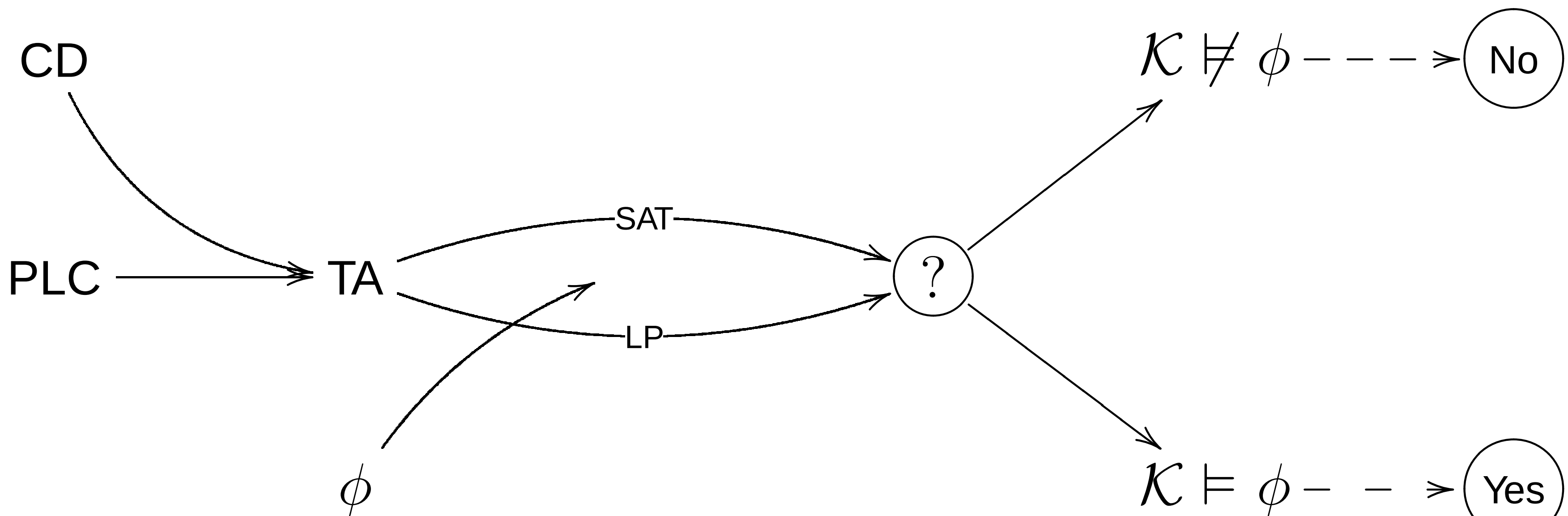
with $T$ transitions on $A$ events of $S$ states in $X$ clocks.

- Many inequalities.
- 9 Fischer: 18167, i.e. 16895 atoms in 42669 clauses.

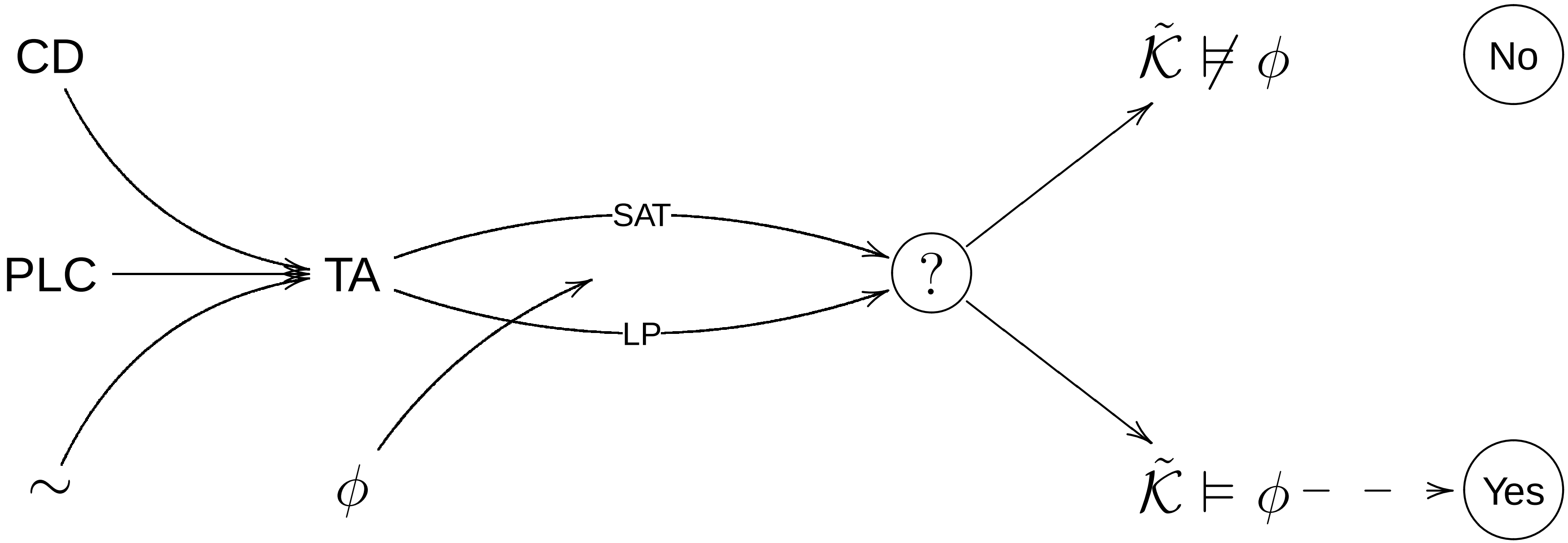
# Extensions (II)

- Early projection of interpolant features.
- *A posteriori* proof-generation by reconstruction.
- Refine interpolants from different abstraction refinement cycles.
- Tableaux-based interpolation?
- (Monadic) Second Order rather than Bounded Model Checking.

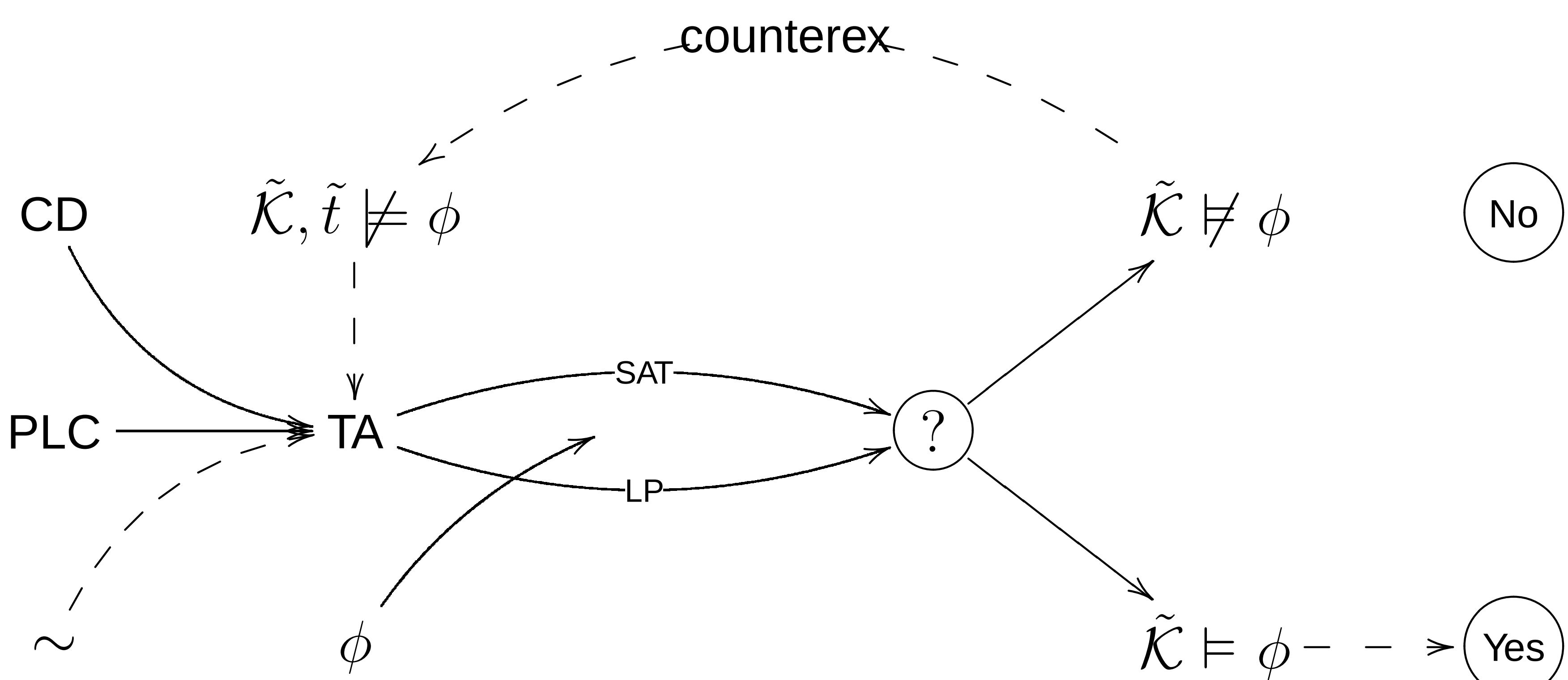
# Architecture



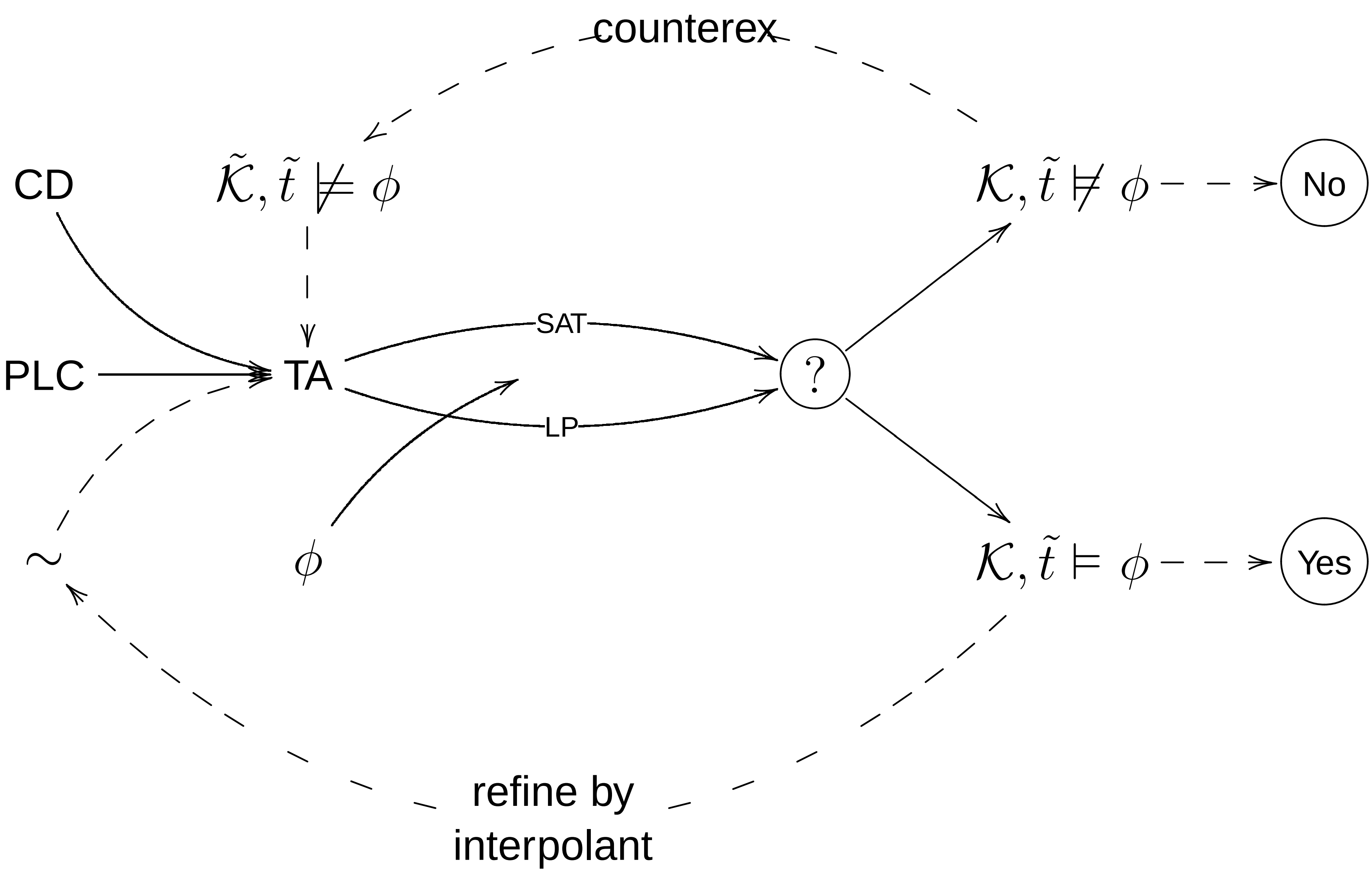
# Abstr. Architecture



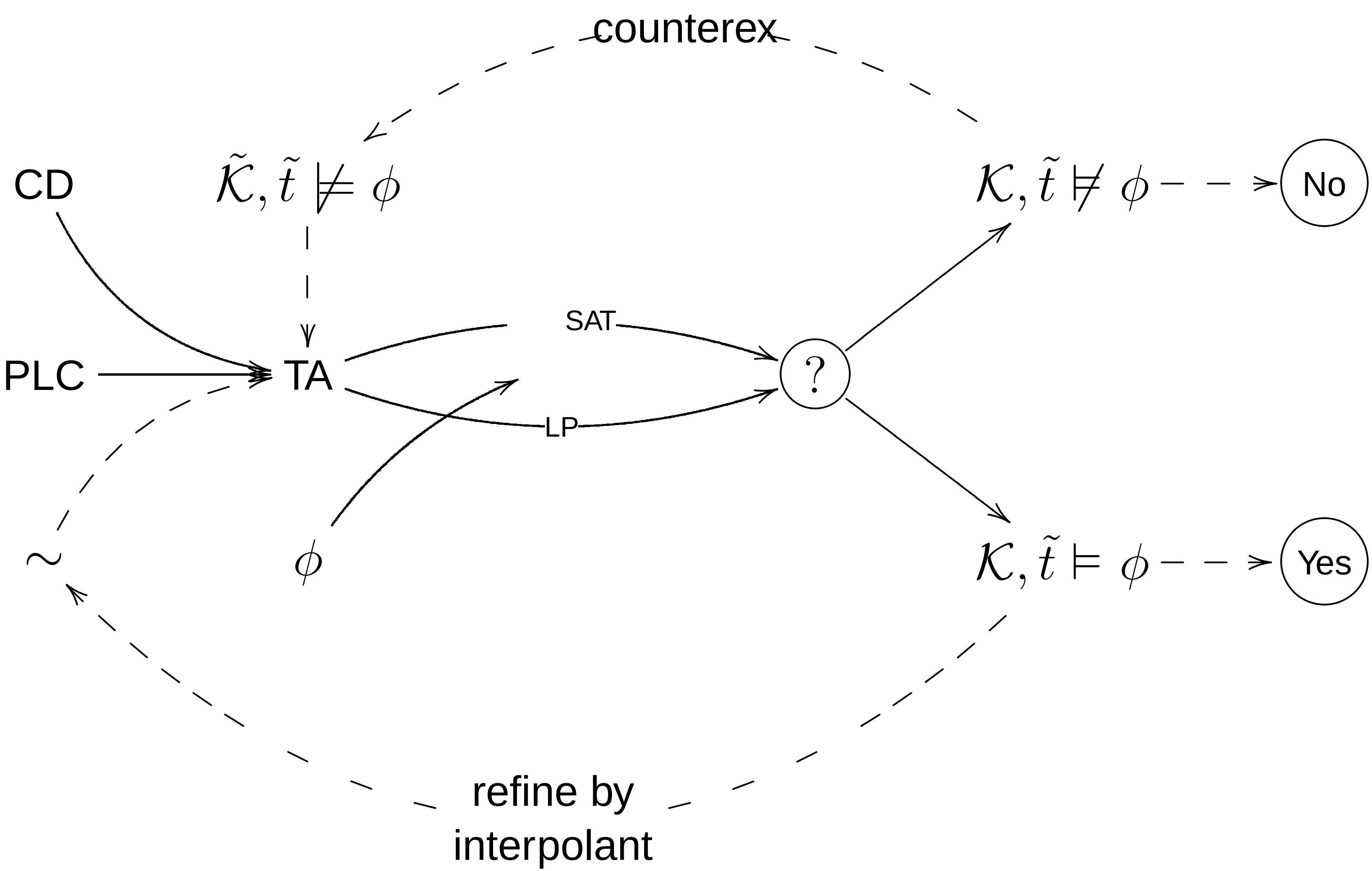
# Abstr. Architecture



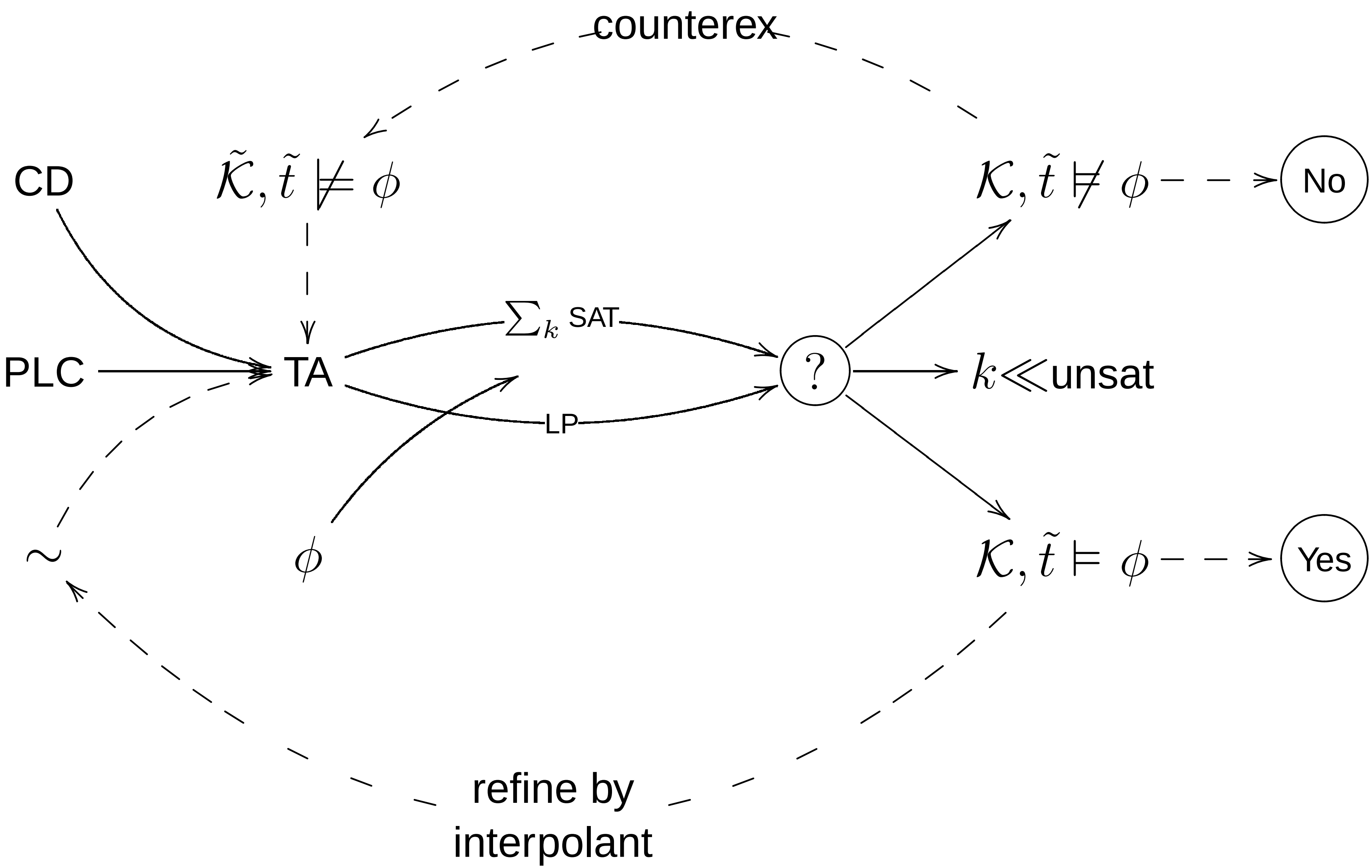
# Abstr. Refin. Architecture



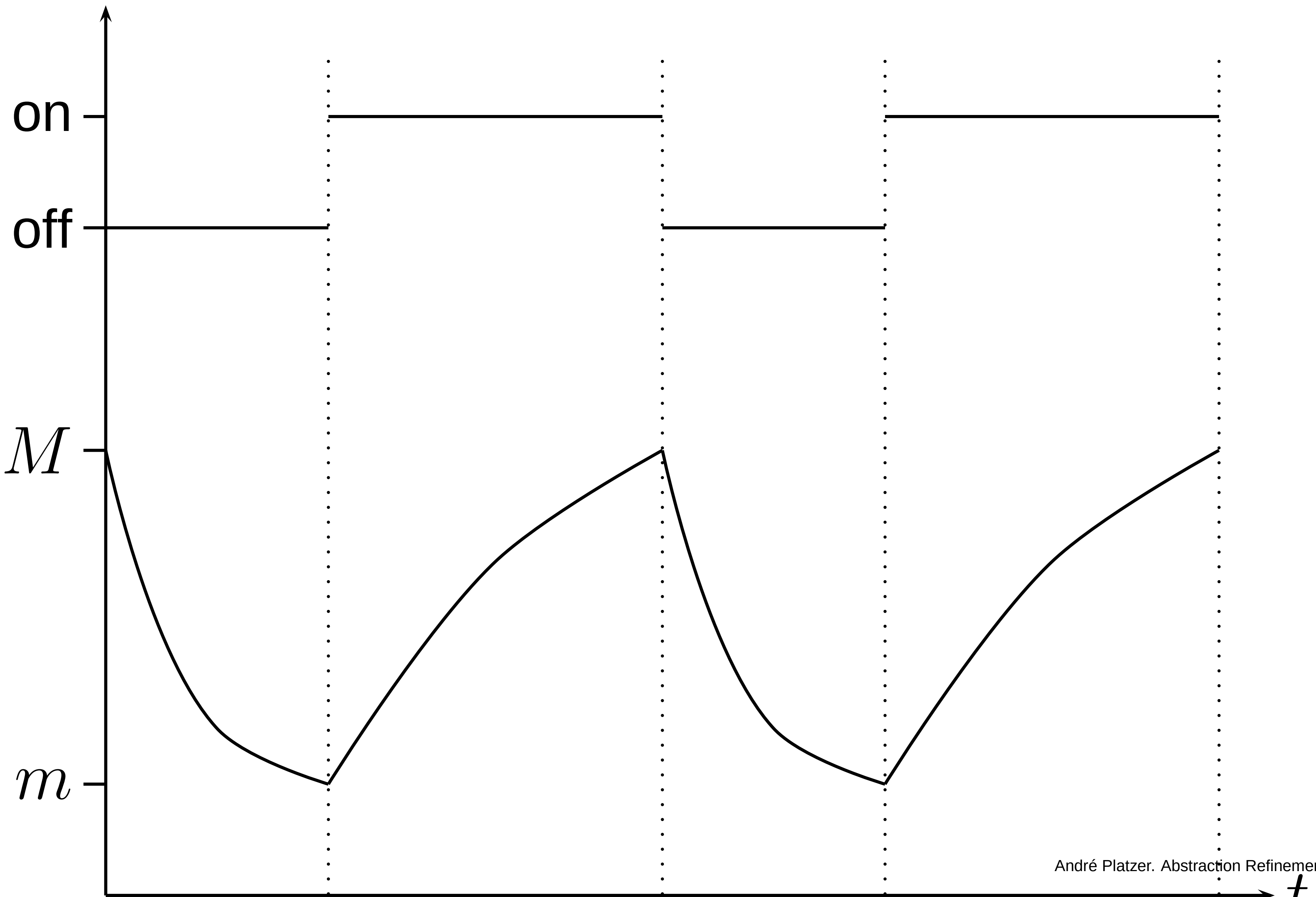
# Abstr. Refin. Architecture



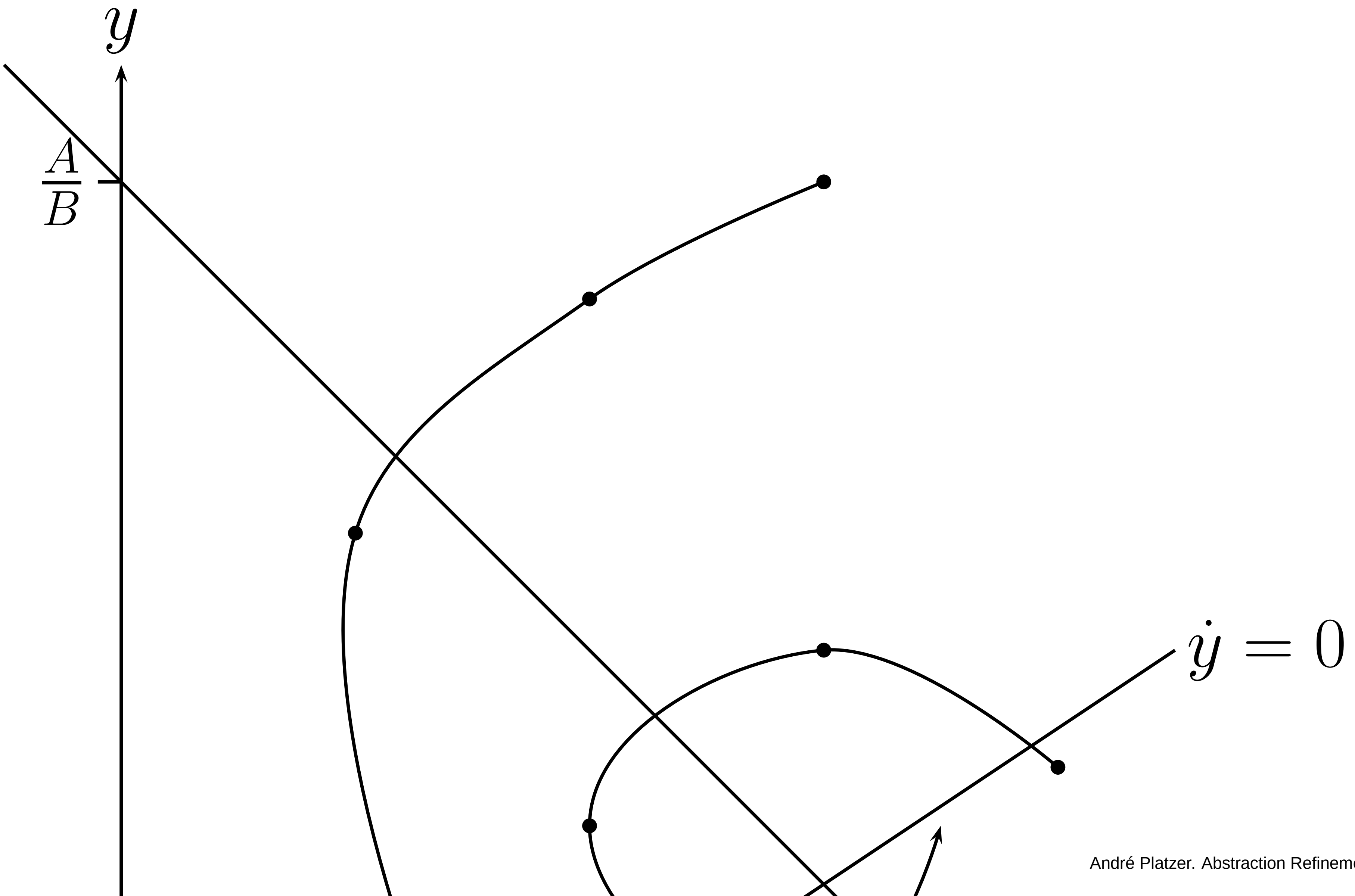
# Abstr. Refin. Architecture



# Heating System



# Predator-Prey



# References

HENZINGER, THOMAS A., JHALA, RANJIT, MAJUMDAR, RUPAK, & MCMILLAN, KENNETH L. 2004. Abstractions from proofs. *Pages 232–244 of: Proceedings of the 31st ACM SIGPLAN-SIGACT symposium on Principles of programming languages*. ACM Press.

MCMILLAN, KENNETH L. 2003. Interpolation and SAT-Based Model Checking. *Pages 1–13 of:* JR., WARREN A. HUNT, & SOMENZI, FABIO (eds), *CAV*. Lecture Notes in Computer Science, vol. 2725. Springer.