Heterogeneous Models of Computation: An Abstract Algebra Approach

EE249 Lecture

Taken from

Roberto Passerone PhD Thesis

Objectives

- Provide the foundation to represent different semantic domains for the Metropolis metamodel
- Study the problem of *heterogeneous interaction*
- Formalize concepts such as abstraction and refinement

An Example of Interaction

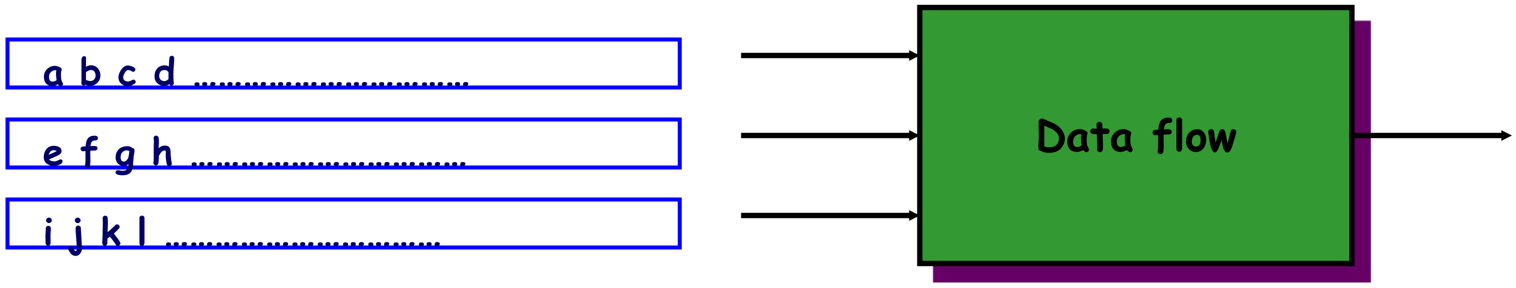
- Combine a synchronous model with a dataflow model
- Synchronous model
  - Total order of event
- Data flow model
  - Partial order of events
- Discrete Time model
  - Metric order of events

An Example of Heterogeneous Interaction

- The interaction is derived from a common refinement of the heterogeneous models
- The resulting interaction depends on the particular refinements employed
- Our objective is to derive the consequences of the interaction at the higher levels of abstraction

Data Flow Model

- Assume signals take values from a set V
- Each signal is a sequence from V (an element of V*)
- Let A be the set of signals
- One behavior is a function
  - $f : A \to V^*$
- A data-flow agent is a set of those behaviors

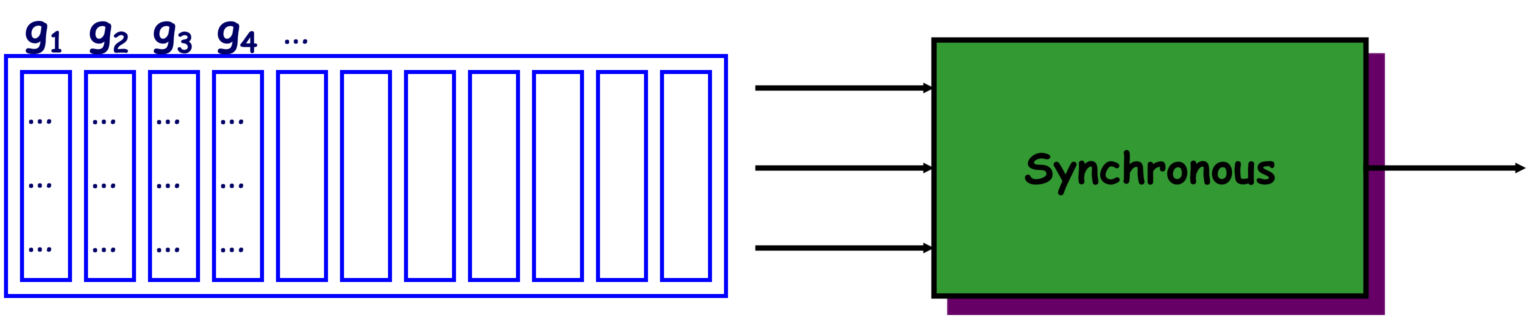


Synchronous Model

- Signals are again sequences from V (elements of V*)

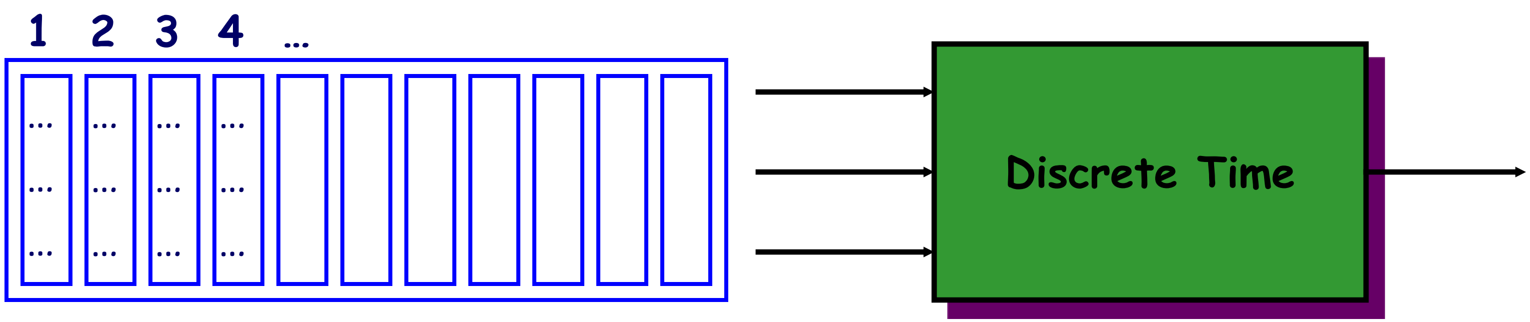
… But are synchronized

- One element of the sequence is $g : A \to V$
- One behavior is a sequence of those functions
  - $\langle g_i \rangle \in (A \to V)^*$
- A synchronous agent is a set of those sequences

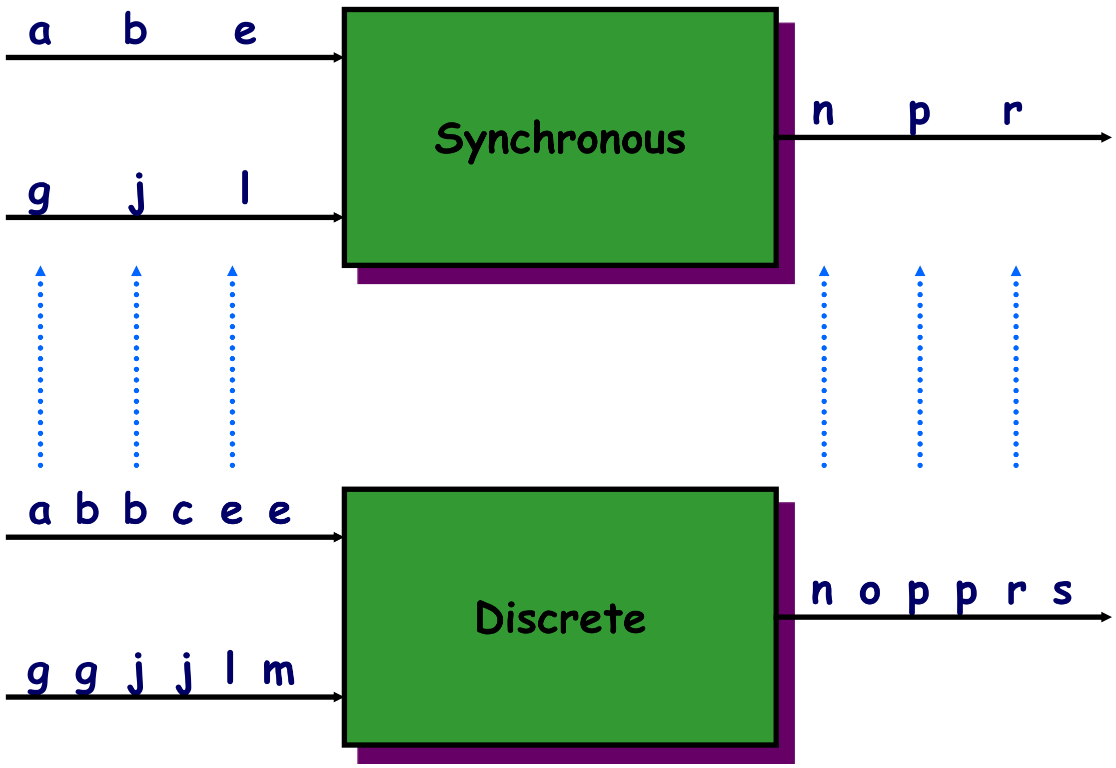


Discrete Time Model

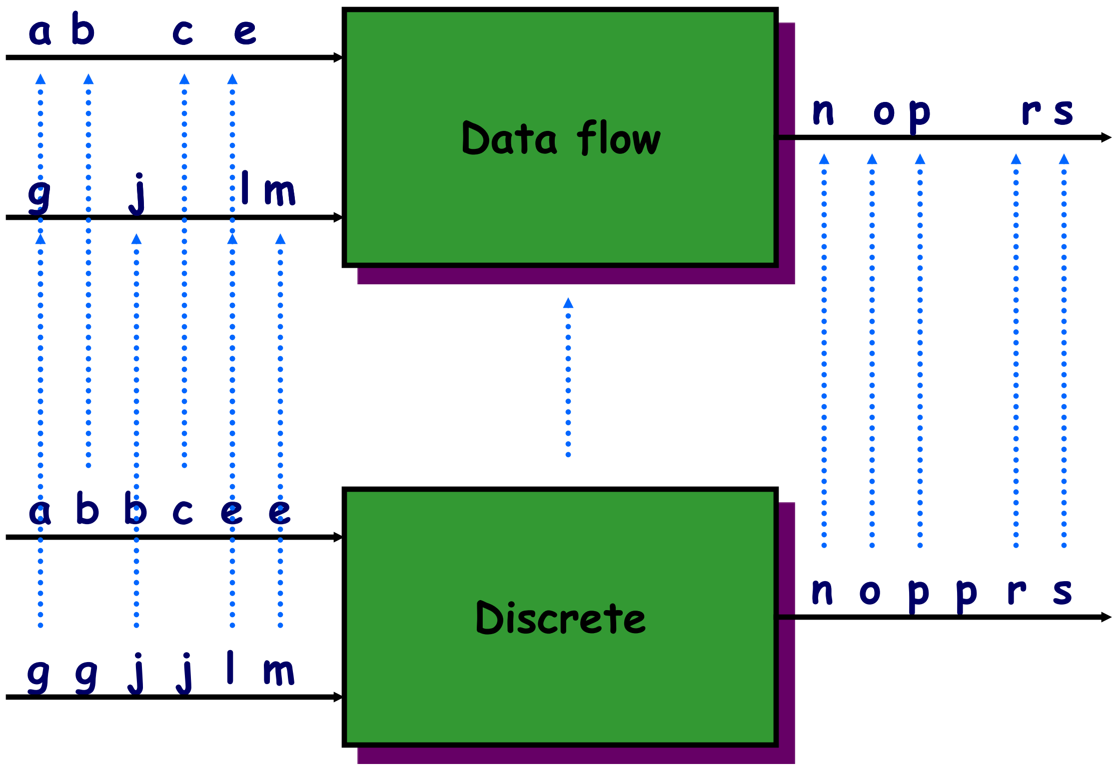
- Assume time is represented by the positive integers N
- Then define a behavior
  - h: N → (A → V)
- A discrete time agent is a set of those functions



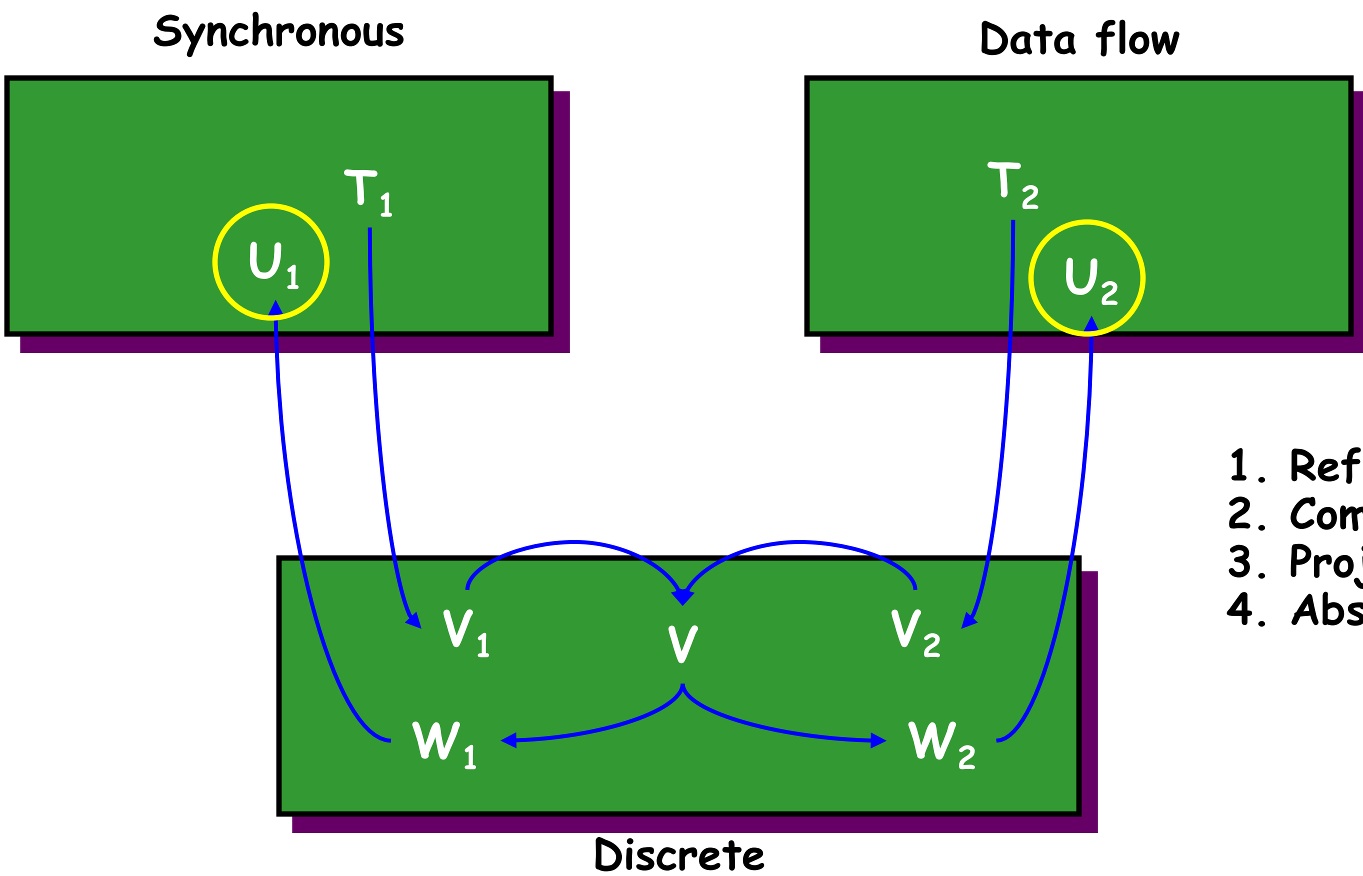
Discrete to Synchronous Abstraction



Discrete to Data Flow Abstraction



Interaction Propagation

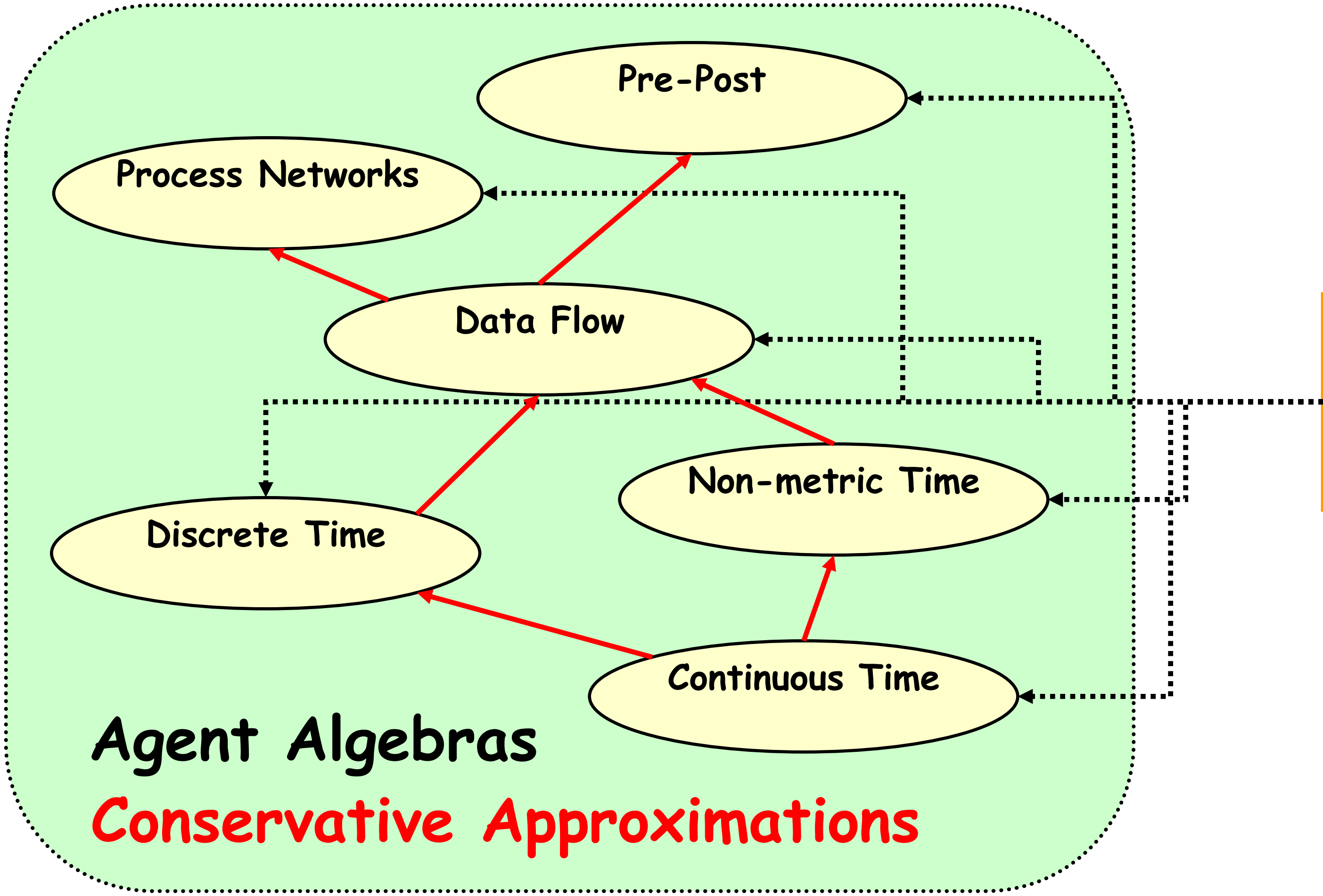


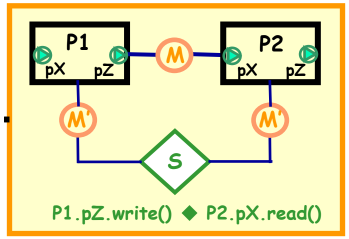
1. Refinement
2. Composition
3. Projection
4. Abstraction

Objectives

- Provide a semantic foundations for integrating different models of computation
  - Independent of the design language
- Maximize flexibility for using different levels of abstraction
  - For different parts of the design
  - At different stages of the design process
  - For different kinds of analysis
- Support many forms of abstraction
  - Model of computation (model of time, synchronization, etc.)
  - Scoping
  - Structure (hierarchy)

Overview





Meta Model

Domain of agents with operations: projection, renaming and composition

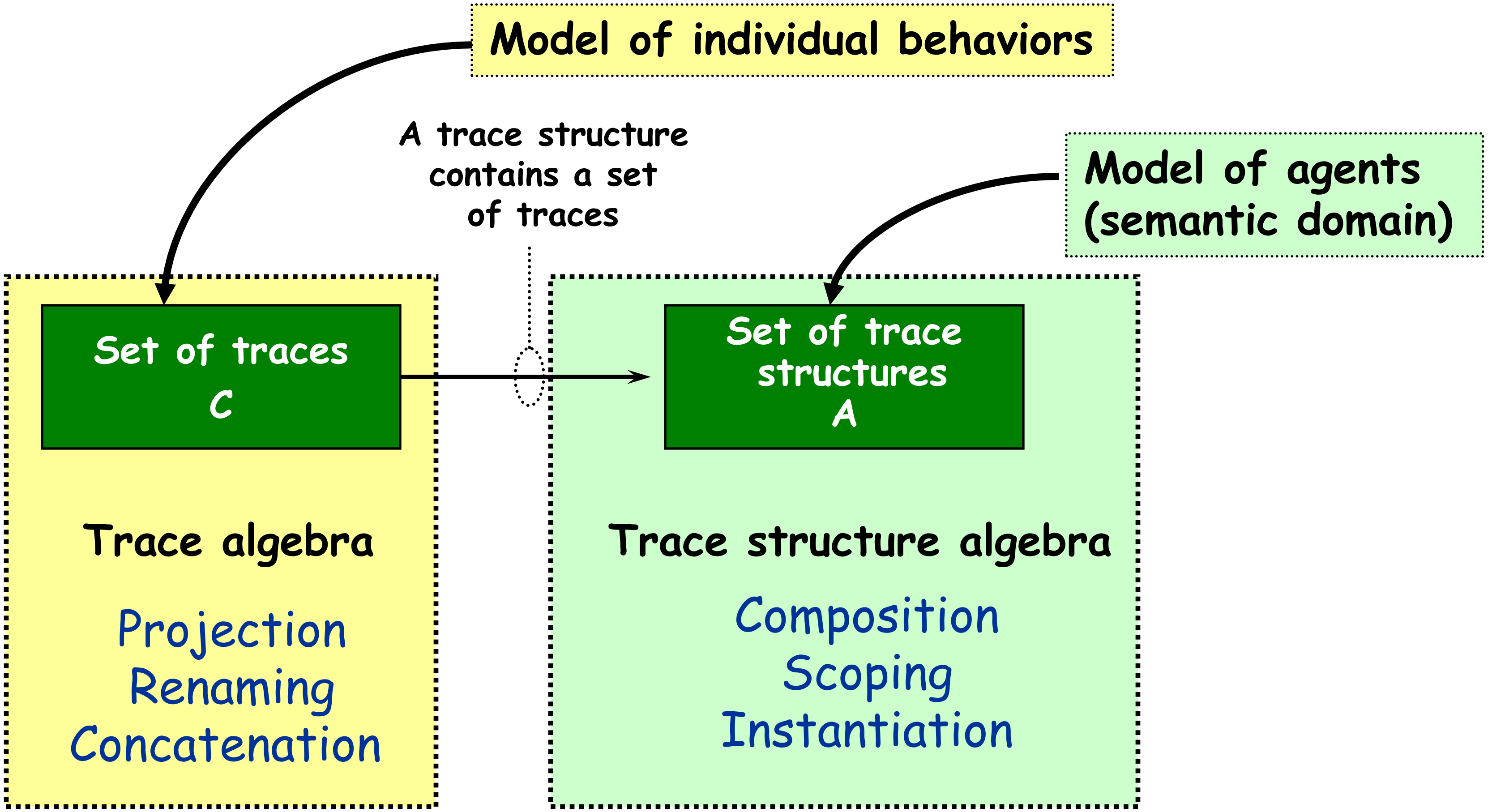
Scope

- Concentrate on
  - Natural semantic domains (sets of agents)
  - Relations and functions over semantic domains
  - Relationships between semantic domains and their relations and functions
- Defer worrying about specific abstract syntaxes and semantic functions
  - Convenient for manual, formal reasoning
  - De-emphasizing executable and finitely-representable models (for now)

Agents and Behaviors

- For each model of computation we always distinguish between
  - the domain of individual behaviors
  - the domain of agents
- For different models of computation individual behaviors can be very different mathematical objects
  - We always call these objects traces
  - The nature of the elements of the carrier is irrelevant!
- An agent is primarily a set P of traces
  - We call them trace structures
  - Also includes the signature: $T = (\gamma, P)$

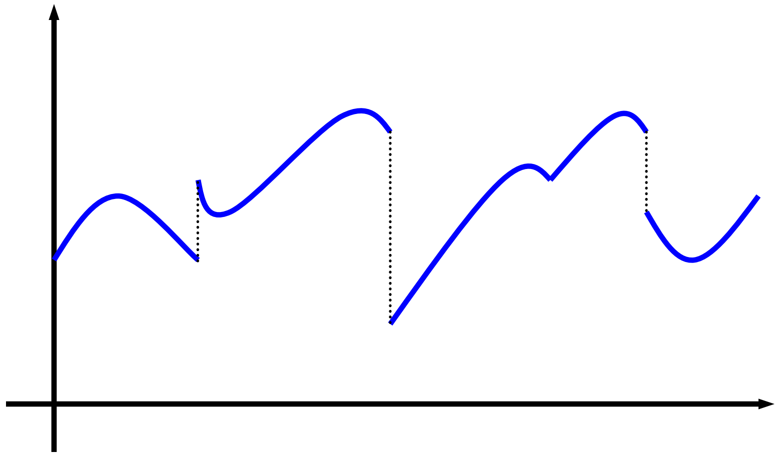
Trace and Trace Structure Algebras



Essential Elements

- Must be able to name elements of the model
  - Variables, actions, signals, states
  - We do not distinguish among them and refer to them collectively as a set of signals W
- Each agent has an alphabet and a signature
  - Alphabet: $A \subseteq W$
  - Signature: $\gamma = A$, $\gamma = (I, O)$, etc.
- The operations on traces and trace structures must satisfy certain axioms
  - The axioms formalize the intuitive meaning of the operations
  - They also provide hypothesis used in proving theorems
  - Trade-off between generality and structure

Metric Time Traces



$$\gamma = (V_R, V_N, M_I, M_O)$$

$$x = (\gamma, \delta, f)$$

$$f(v) = [0, \delta] \text{ -> } R$$
$$f(n) = [0, \delta] \text{ -> } N$$
$$f(a) = [0, \delta] \text{ -> } \{0, 1\}$$

- Model time as a metric space
  - Can talk about the difference in time between points in the behavior in quantitative terms
  - Able to specify timing constraints in quantitative terms
- Able to represent continuous as well as discrete behavior
- Projection and renaming easily defined on the functions

Metric Time Model: Traces

- A trace x models one execution of a hybrid system:
- Signature γ = (
  V_R: real valued var's,
  V_N: integer valued var's,
  M_I: input actions,
  M_O: output actions)
- The alphabet A of x is the union of the components of γ
- δ is a non-negative real number
  - Length (in time) of x
  - Can be infinity

- f gives values as a function of time:
  f: V_R --> [0, δ] --> R,
  f: V_N --> [0, δ] --> N,
  f: M_I --> [0, δ] --> {0, 1},
  f: M_O --> [0, δ] --> {0, 1}.

Metric Time Model: Operations on Traces

- Let $x' = proj(B)(x)$
  - represents scoping
  - B is a subset of A
  - γ' and f' are restricted to variables and actions in B
  - $\delta' = \delta$
- Let $x' = rename(r)(x)$
  - represents instantiation
  - r is a one-to-one function with domain A
  - variables and actions in γ' and f' are renamed by r
  - $\delta' = \delta$
- Let $x'' = x \cdot x'$ (concatenation)
  - represents sequential composition
  - $\gamma' = \gamma$, δ is finite, and end of x matches beginning of x'
  - $\gamma'' = \gamma$
  - $\delta'' = \delta + \delta'$
  - $f''(v, t)$ is equal to
    $f(v, t)$ for $t \leq \delta$
    $f'(v, t - d)$ for $t \geq \delta$

Metric Time Model: Trace Structures

- A trace structure T = (γ, P) models a process or an agent of a hybrid system
  - P is a set of traces with signature γ

Traits:

- T refines T' if $P \subseteq P'$
- Natural model for physical components (such as those described with differential equations, possibly with discrete control variables)
- Too detailed for many other aspects of embedded systems
- Not a finite representation
  - Finite representations, synthesis and verifications algorithms are clearly important, but not a focus of this class
- Trace structures constructed the same way for any trace algebra

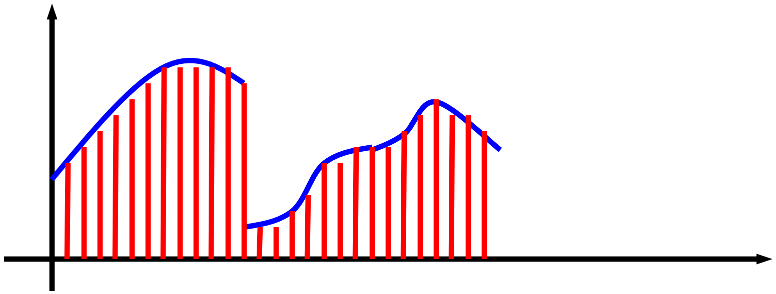
Metric Time Model: Operations on Trace Structures

- Let T' = proj(B)(T)
  - B is a subset of A
  - γ' is restricted to variables and actions in B
  - P' = proj(B)(P)
- Let T' = rename(r)(T)
  - r is a one-to-one function with domain A
  - variables and actions in γ' are renamed by r
  - P' = rename(r)(P)
- Let T'' = T || T' (par. comp.)
  - γ'' combines γ and γ'
  - P'' maximal set s.t.

    P = proj(A)(P'')

    P' = proj(A')(P'')
- Let x'' = x · x' (seq. comp.)
  - $\gamma' = \gamma$
  - P'' = P · P' (roughly)

Non-metric Time Traces



$\gamma = (V_R, V_N, M_I, M_O)$

$x = (\gamma, L)$

$m(t) = V_R \text{ -> } R$

$V_N \text{ -> } N$

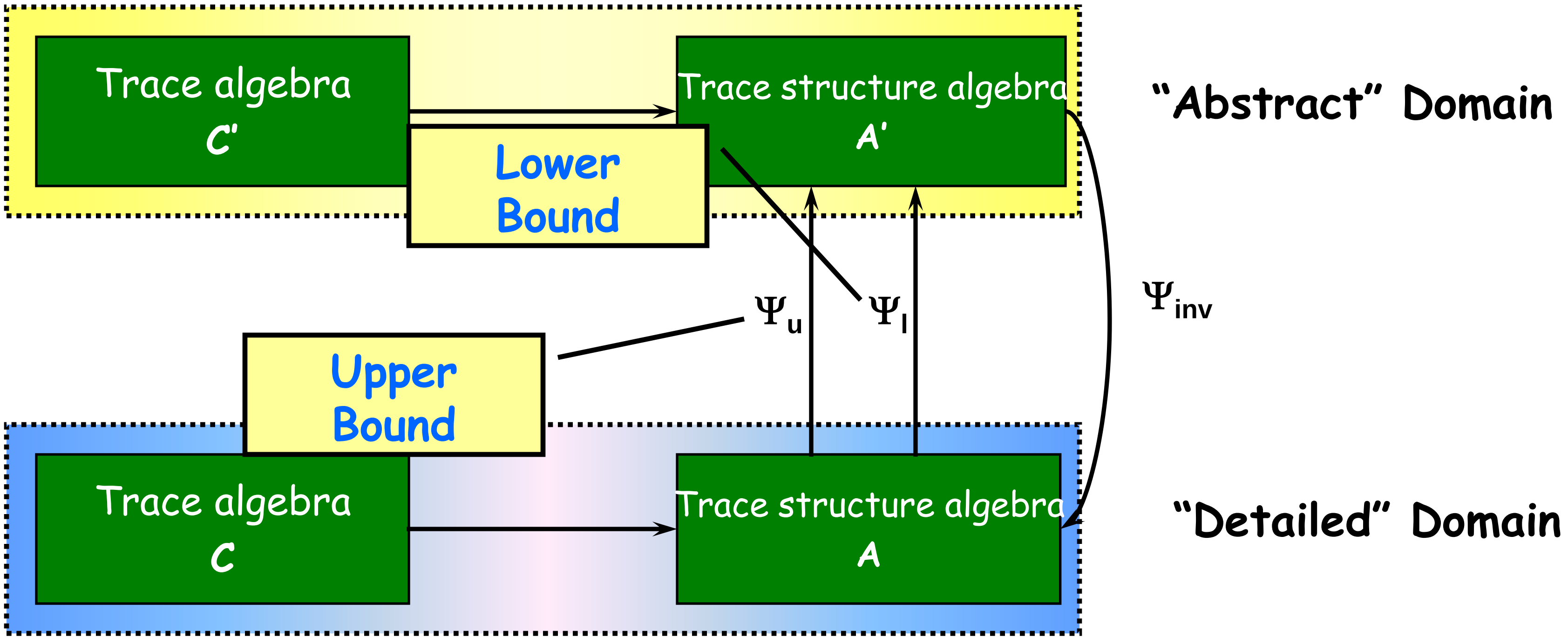
$M \text{ -> } \{ 0, 1 \}$

- **Model time as a non-metric space**
  - Can only talk about precedence in time (including dense time)
- **Based on Totally Ordered Multi-Sets**
  - Totally ordered vertex set V
  - Labeling function μ from the vertex set V to a set of actions Σ
  - We do not distinguish isomorphic vertex sets

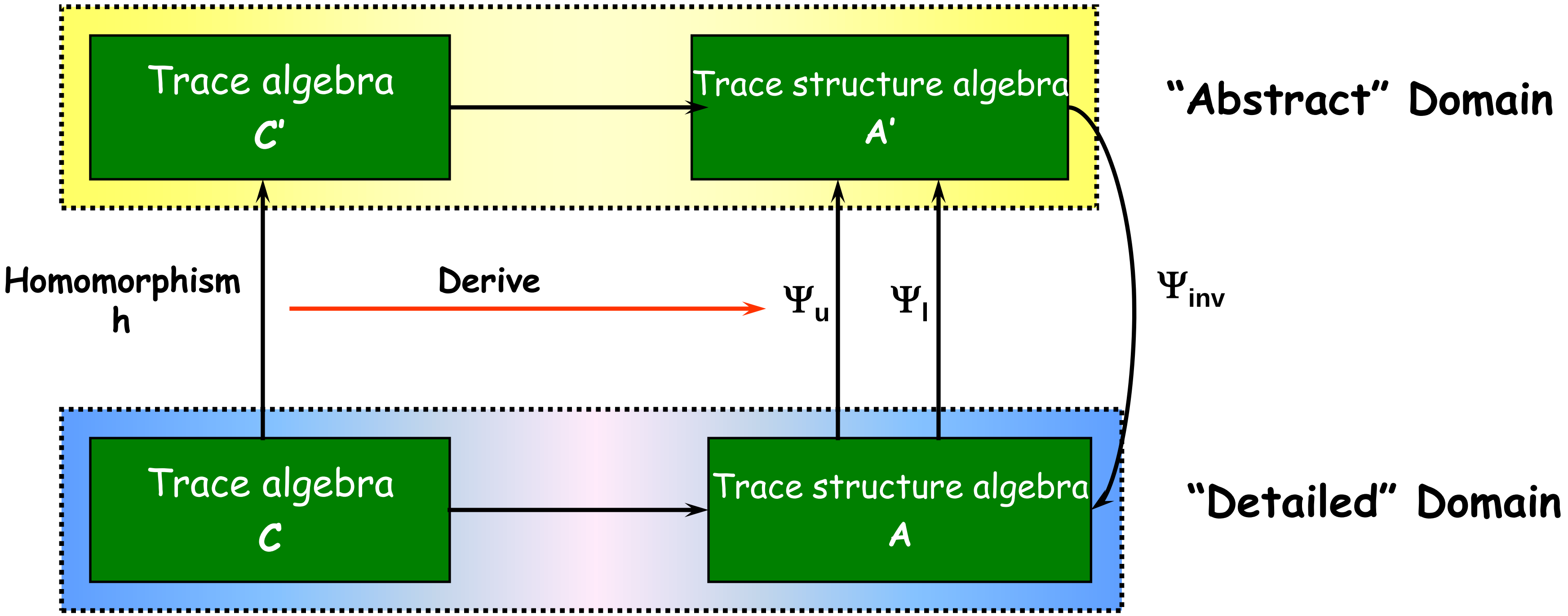
Relationships between Semantic Domains

- Each semantic domain has a refinement order
  - Based on trace containment
  - $T_1 \subseteq T_2$ means T_1 is a refinement of T_2
  - Guiding intuition: $T_1 \subseteq T_2$ means T_1 can be substituted for T_2
- Abstraction mapping
  - If a function H between semantic domains is monotonic, detailed implies abstract: If $T_1 \subseteq T_2$ then $H(T_1) \subseteq H(T_2)$
  - Analogy for real numbers r and s: If $r \leq s$ then $\lfloor r \rfloor \leq \lfloor s \rfloor$
- Conservative approximations
  - A pair of functions $\Psi = (\Psi_l, \Psi_u)$ is a *conservative approximation* if $\Psi_u(T_1) \subseteq \Psi_l(T_2)$ implies $T_1 \subseteq T_2$
  - Analogy: $\lceil r \rceil \leq \lfloor s \rfloor$ implies $r \leq s$
  - Abstract implies detailed

Trace and Trace Structure Algebras



Deriving Conservative Approximations

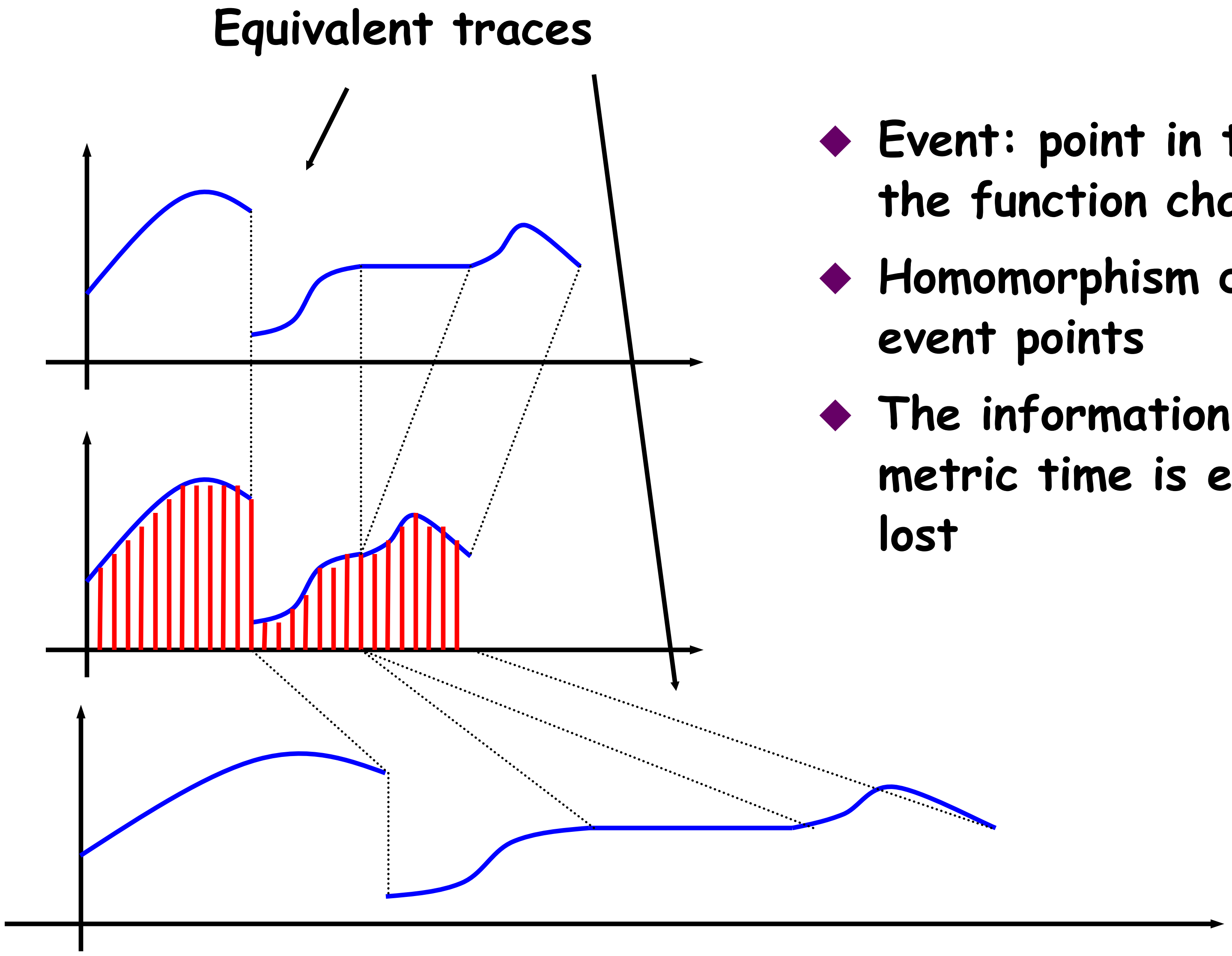


Homomorphism: mapping that commutes with the operations of projection, renaming and concatenation on traces

Homomorphism

- From metric to non-metric
  - Must define a notion of event in the metric model
  - Must define how to construct the corresponding vertex set
- From non-metric to pre-post
  - Simply remove the intermediate steps and keep only the end-points

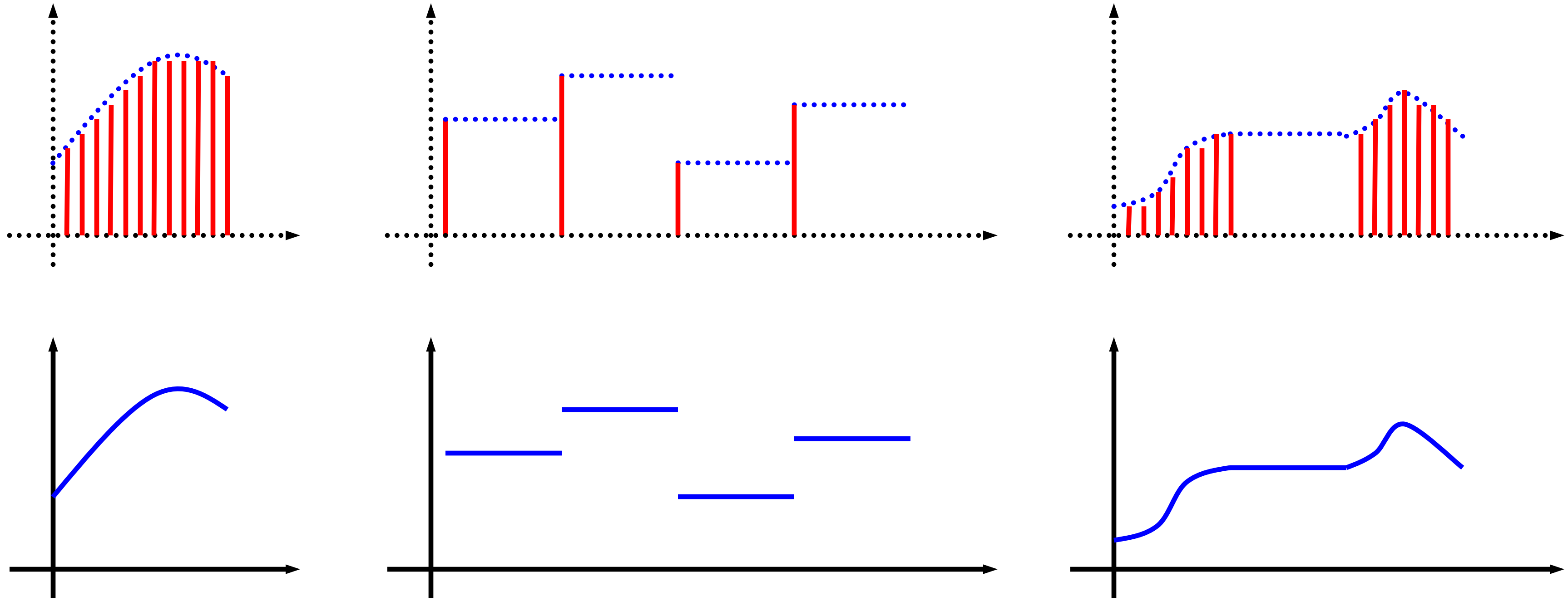
Metric to Non-Metric Traces



- Event: point in time where the function changes value
- Homomorphism discards non-event points
- The information about metric time is effectively lost

From Metric to Non-metric Time

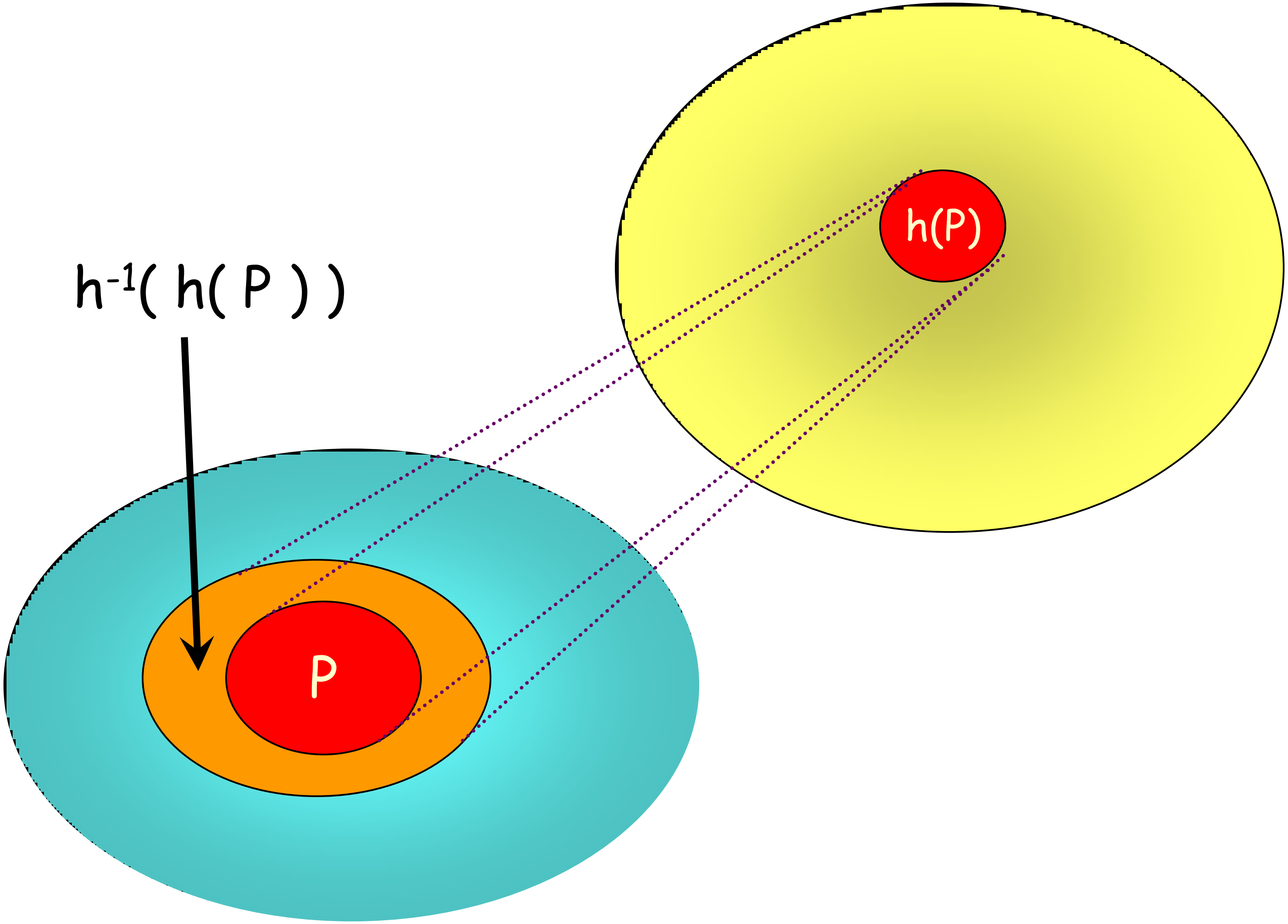
- f is stable at t_0 if there is $\varepsilon > 0$ such that f is constant on $[\, t_0 - \varepsilon \,,\, t_0 \,]$
- f has an event at t_0 if it is not stable
- Vertex Set $V = \{ \, t_0 \, / \, f$ has an event at $t_0 \, \}$



Building the Upper Bound

- Let P be a set of traces, and consider the natural extension to sets h(P) of h
- Clearly $P \subseteq h^{-1}(h(P))$
  - Because h is many-to-one
  - This indeed is an upper bound!
  - Equality holds if h is one-to-one
- Hence define
  - $\Psi_u(T) = (\gamma, h(P))$

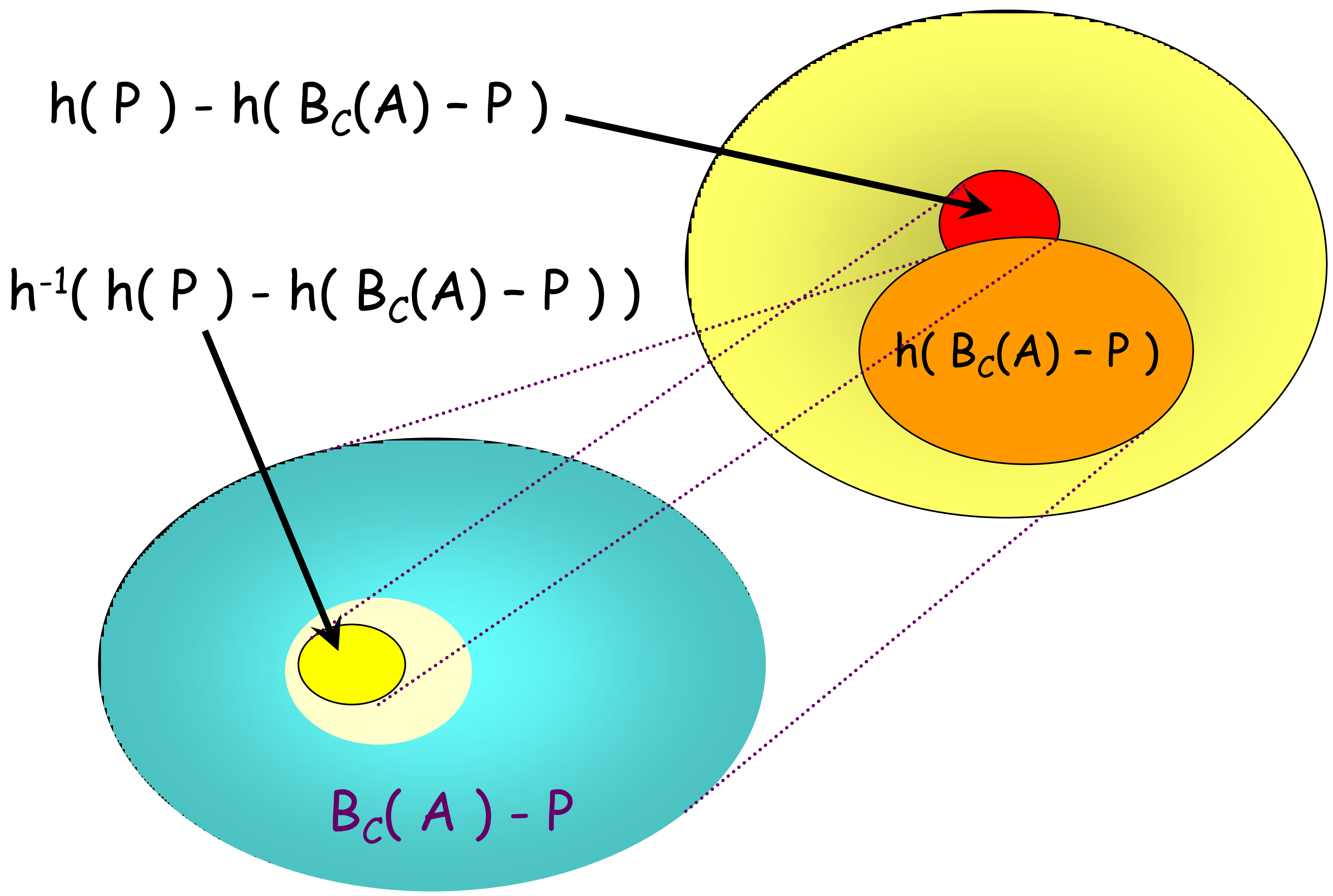
Building the Upper Bound



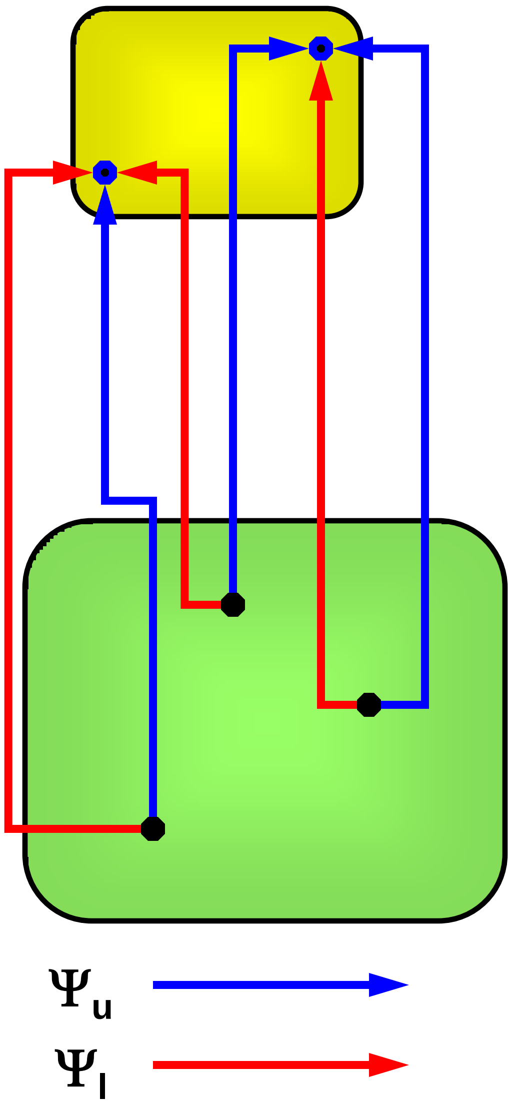
Building the Lower Bound

- We want $P \supseteq h^{-1}$(lb of P)
- If x is not in P, then h(x) should not be in the lower bound of P
- Hence define
  - $\Psi_l(T) = h(P) - h(B_C(A) - P)$
- There is a tighter lower bound

Building the Lower Bound



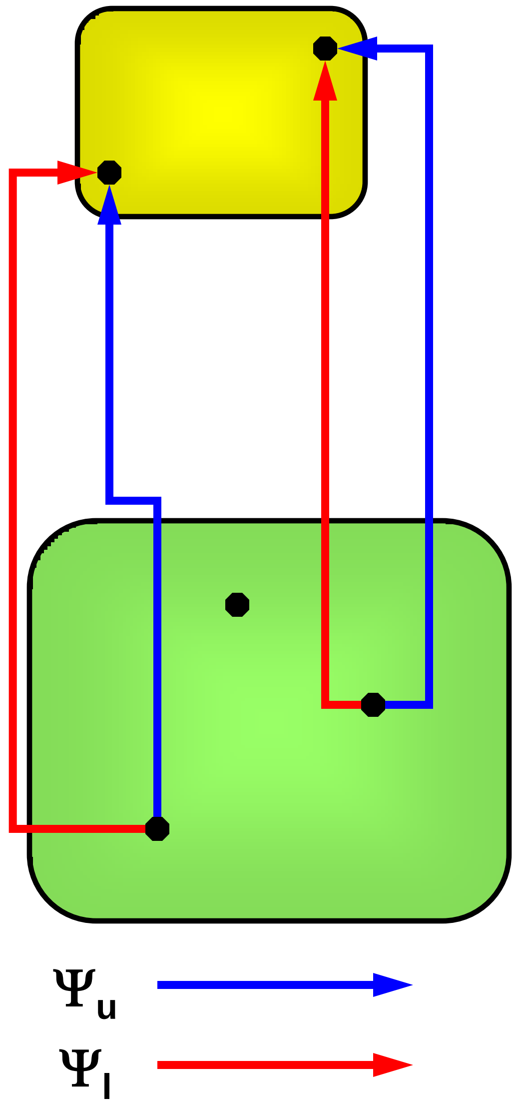
Conservative Approximations: Inverses



- Apply Ψ_u
- Apply Ψ_l
- Consider T such that

$$\Psi_u(\ T\ ) = \Psi_l(\ T\ ) = T'$$

Conservative Approximations: Inverses

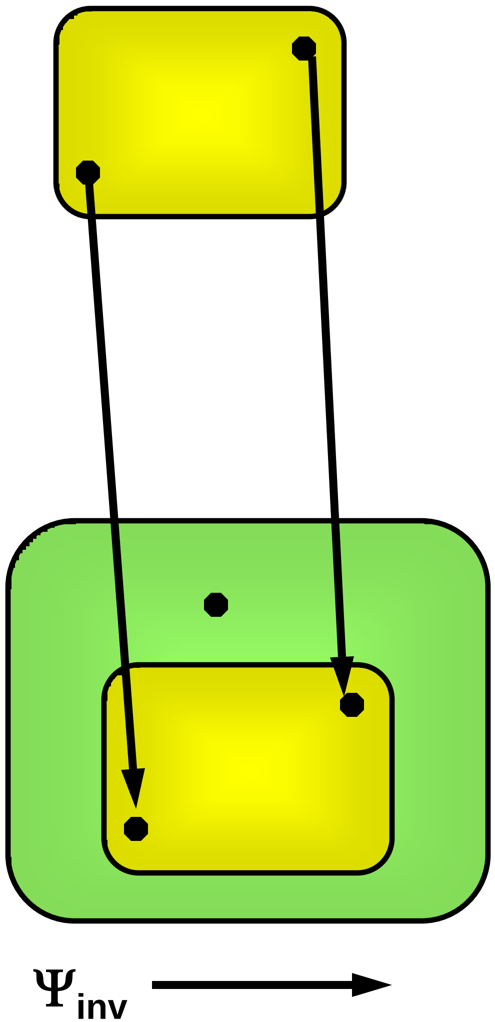


- Apply Ψ_u
- Apply Ψ_l
- Consider T such that

  $\Psi_u(T) = \Psi_l(T) = T'$

- Then $\Psi_{inv}(T') = T$

Conservative Approximations: Inverses

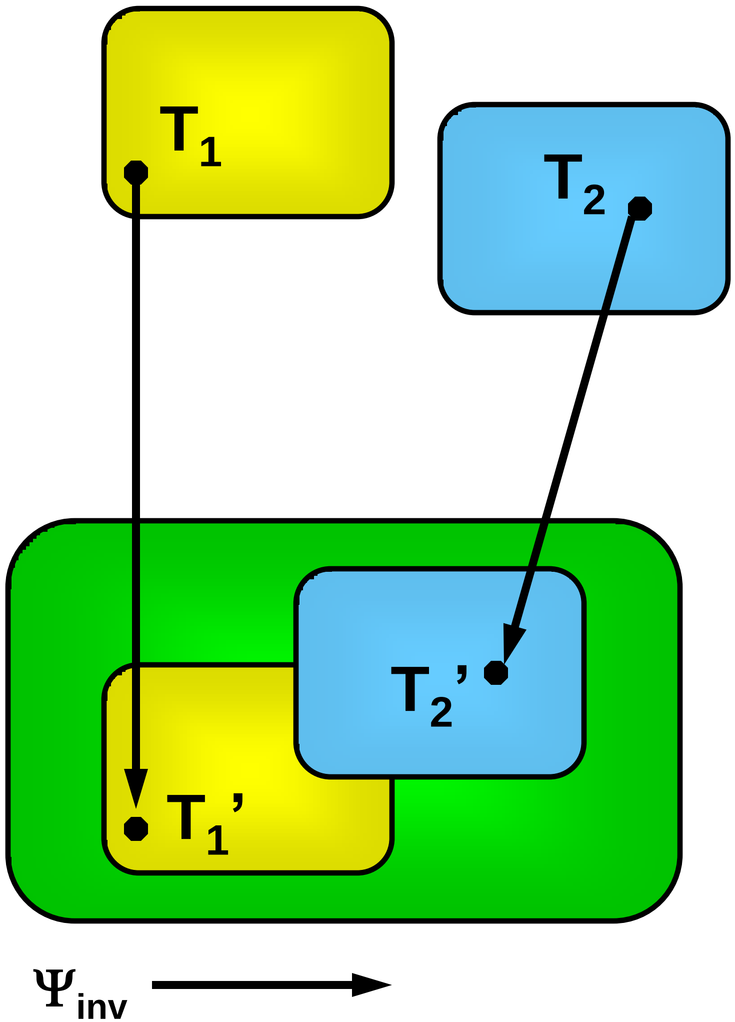


- Apply Ψ_u
- Apply Ψ_l
- Consider T such that

  $\Psi_u(T) = \Psi_l(T) = T'$

- Then $\Psi_{inv}(T') = T$
- Can be used to embed high-level model in low level

Combining MoCs



Want to compose T_1 and T_2 from different trace structure algebras

- Construct a third, more detailed trace algebra, with homomorphisms to the other two
- Construct a third trace structure algebra
- Construct cons. approximations and their inverses
- Map T_1 and T_2 to T_1' and T_2' in the third trace structure algebra
- Compose T_1' and T_2'

Conclusions

- Semantic foundations for the Metropolis meta-model
- All models of computation of importance "reside" in a unified framework
  - They may be better understood and optimized
- Trace Algebra used as the underlying mathematical machinery
  - Showed how to formalize a semantic domain for several models of computation
- Conservative approximations and their inverses used to relate different models