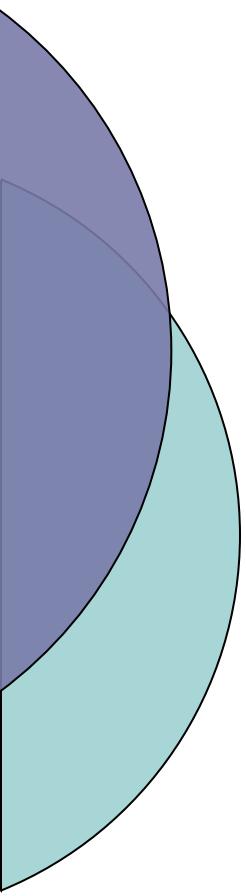


Hyperproperties

Michael Clarkson
Cornell University

15th International School on Foundations of Security Analysis and Design
University Residential Center of Bertinoro, Italy
September 3, 2015

Security Policies Today

Confidentiality

"Protection of assets from unauthorized disclosure"

Integrity

"Protection of assets from unauthorized modification"

Availability

"Protection of assets from loss of use"

Confidentiality

- Keep contents of file secret
- Keep value of variable secret
- Keep behavior of system secret
- Keep information about individual secret
- …

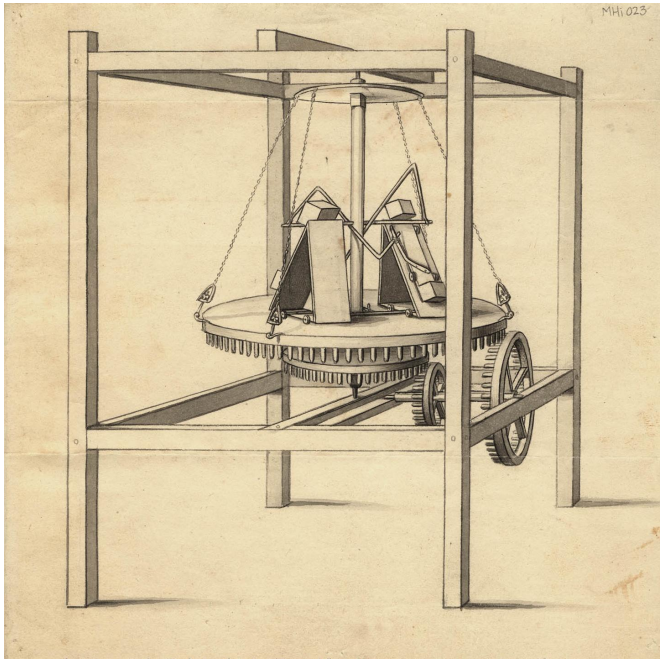
Various areas: access control, information flow, covert channel control, database privacy…

Integrity

- Output is correct according to functional specification
- Resource consumption is bounded
- Data are correct w.r.t. external entities
- Only certain operations are permitted to certain principals
- Information is not corrupted or tainted
- …

Availability

- System must accept inputs periodically
- System must produce output by specified time
- System must process requests fairly
- …



Security Policies Today

- **No formalism** that expresses all of CIA
- **Orthogonal?**
  - e.g., "Alice can't read variable x": C or I or A?
- **Expressively complete?**
- **Complete verification methodologies?**

Formalize and verify any security policy? 

Program Correctness ca. 1970s

- Partial correctness (If program terminates, it produces correct output)
- Termination
- Total correctness (Program terminates and produces correct output)
- Mutual exclusion
- Deadlock freedom
- Starvation freedom

???

Safety and Liveness Properties

Intuition [Lamport 1977]:

Safety:
"Nothing bad happens"

- Partial correctness
  Bad thing: program terminates with incorrect output
- Access control
  Bad thing: subject completes operation without required rights

Liveness:
"Something good happens"

- Termination
  Good thing: termination
- Guaranteed service
  Good thing: service rendered

Properties

Trace: Sequence of execution states

$t = s_0 s_1 \ldots$

Property: Set of infinite traces

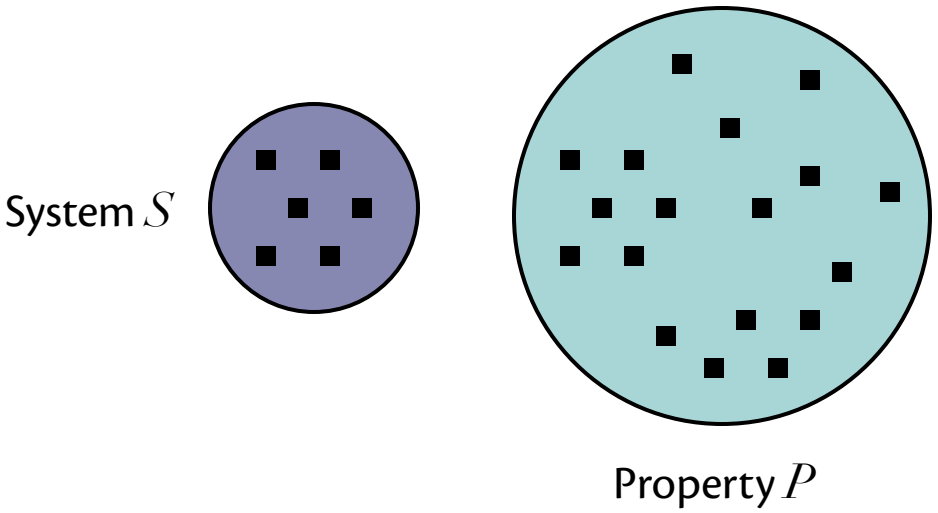
Trace t satisfies property P iff t is an element of P

➔ Satisfaction depends on the trace alone

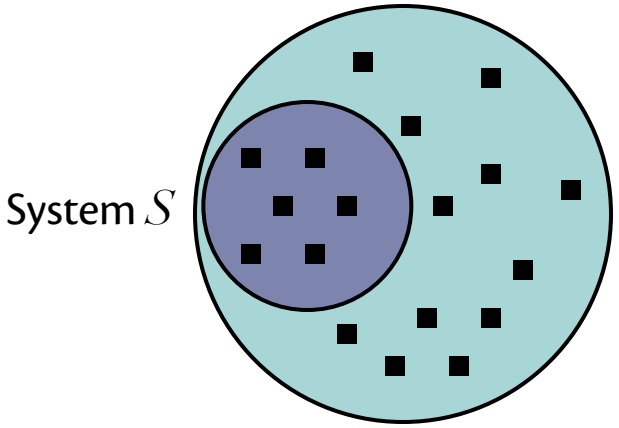
System: Also a set of traces

System S satisfies property P iff all traces of S satisfy P

Properties



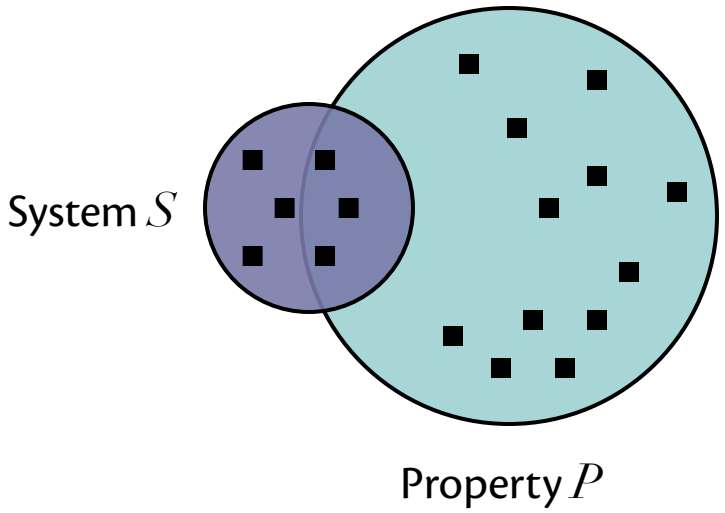
Properties



S **satisfies** P

■ = trace

Properties



S does not satisfy P

■ = trace

Safety and Liveness Properties

Formalized:

Safety property [Lamport 1985]

Bad thing = trace prefix

Liveness property [Alpern and Schneider 1985]

Good thing = trace suffix

Success!

Alpern and Schneider (1985, 1987):

Theorem. *Every property is the intersection of a safety property and a liveness property.*

Theorem. *Safety proved by invariance.*

Theorem. *Liveness proved by well-foundedness.*

Theorem. *Topological characterization:*

Safety = closed sets

Liveness = dense sets

Formalize and verify any property? ✓

Back to Security Policies

Formalize and verify any property? ✓

Formalize and verify any security policy? ✗

$$\text{Security policy} \stackrel{?}{=} \text{Property}$$

Information Flow is not a Property

Secure information flow:

Secret inputs are not leaked to public outputs

p := 1; ✓

p := s; ✗

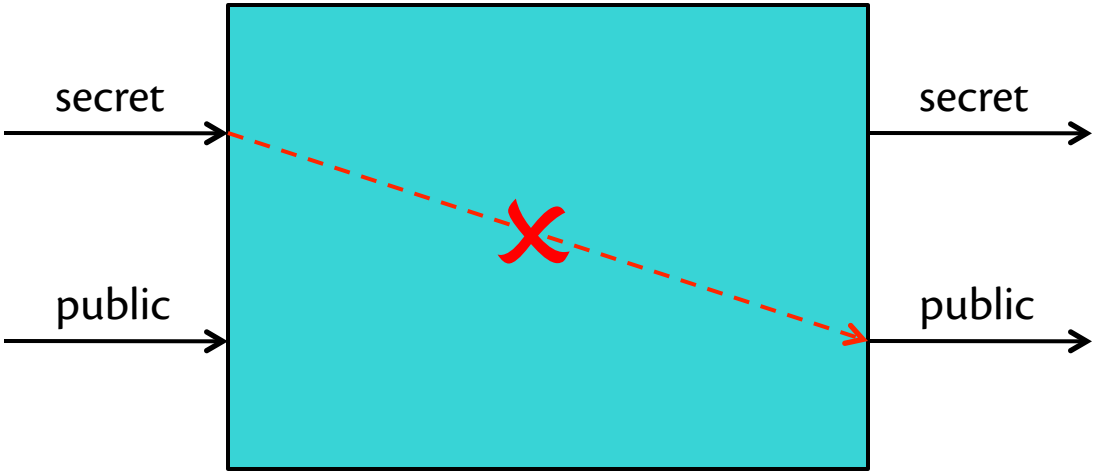
if (s) then p := 1 else p := 0; ✗

if (s) then {consume power} else {don't}; ✗

Information Flow is not a Property

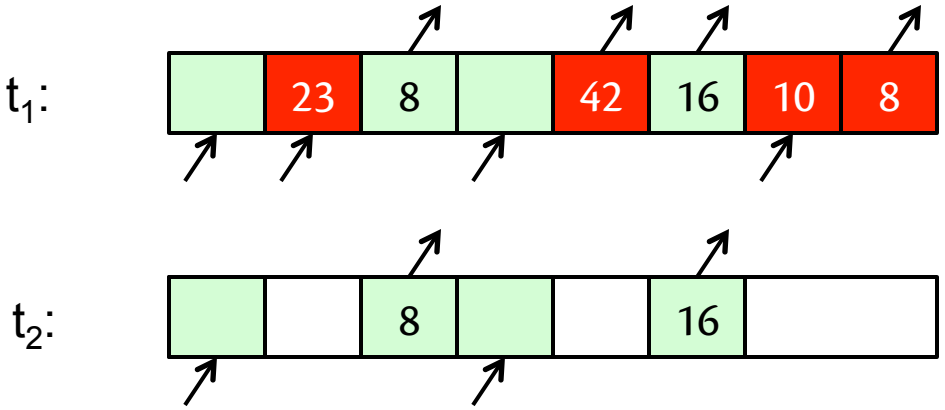
Secure information flow:

Secret inputs are not leaked to public outputs



Information Flow is not a Property

Noninterference [Goguen and Meseguer 1982]: Commands of high security users have no effect on observations of low security users



Not safety!

Satisfaction depends on pairs of traces …so not a property

Service Level Agreements are not Properties

Service level agreement: Acceptable performance of system

Not liveness!

Average response time: Average time, over all executions, to respond to request has given bound

- Satisfaction depends on all traces of system …not a property

Any security policy that stipulates relations among traces is not a property

➔ Need satisfaction to depend on *sets* of traces [McLean 1996]

Hyperproperties

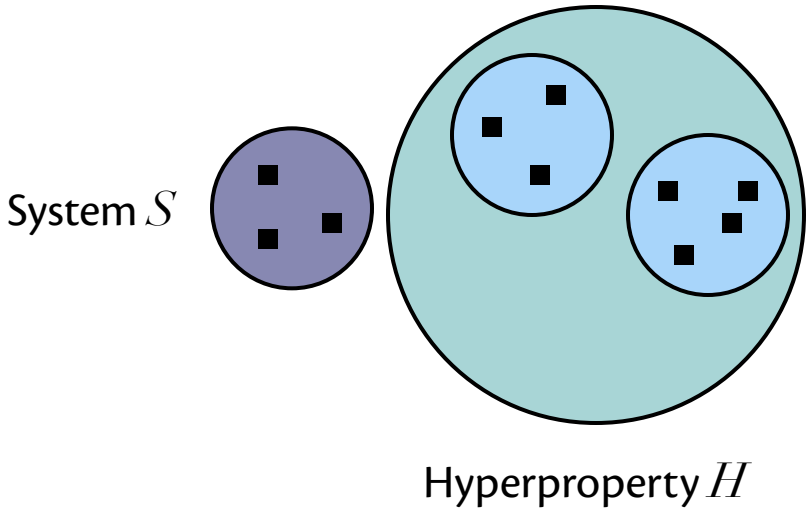
A **hyperproperty** is a set of properties

[Clarkson and Schneider 2008, 2010]

A system S **satisfies** a hyperproperty H
iff S is an element of H

...a hyperproperty specifies exactly the allowed sets of traces

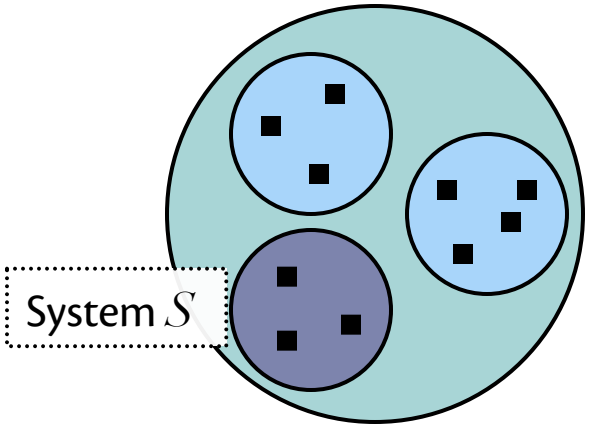
Hyperproperties



S **does not satisfy** H

■ = trace

Hyperproperties



S **satisfies** H

■ = trace

Hyperproperties

Security policies are hyperproperties!

- **Information flow:** Noninterference, relational noninterference, generalized noninterference, observational determinism, self-bisimilarity, probabilistic noninterference, quantitative leakage
- **Service-level agreements:** Mean response time, time service factor, percentage uptime
- …

Noninterference

[Goguen and Meseguer 1982]

Intuition: for every trace, there's another trace with the same low observation but devoid of high inputs

$$SM = \{T \mid T \text{ is a deterministic state machine}\}$$

$$GMNI = \{\ T \text{ in } SM \mid \text{forall } t \text{ in } T, \text{ exists } t' \text{ in } T,$$
$$\text{high-in}(t') = \varepsilon \text{ and } \text{low}(t) = \text{low}(t')\ \}$$

high-in(u) = restriction of u to high input events

low(u) = restriction of u to low events

ε = empty trace

Generalized Noninterference

[McCullough 1987, McLean 1996] (simplification)

Intuition: for every high input sequence and every low observation, there's a trace that combines them

GNI = { T | forall t1,t2 in T, exists t3 in T,
high-in(t3) = high-in(t1)
and low(t3) = low(t2) }

Observational Determinism

[McLean 1992, Roscoe 1995, Zdancewic and Myers 2003]

Intuition: system appears to be deterministic function of low inputs

$$OD = \{\ T \mid \text{forall } t,t' \text{ in } T,\ \text{low-in}(t[0]) = \text{low-in}(t'[0]) \text{ implies } \text{low}(t) = \text{low}(t')\}$$

Mean Response Time

Intuition: mean response time over all executions is less than 1 second

$$RT = \{ T \mid mean(\{respTimes(t) \mid t \text{ in } T\}) <= 1 \}$$

respTimes(t) = set of response times from request-response events in t

Beyond Hyperproperties?

- Security policies are *predicates* on systems
- An *extension* of a predicate is the set of all its models
- A hyperproperty is the set of all systems that satisfy a predicate, i.e., the extension of the predicate

➔ Hyperproperties are expressively complete

(for predicates, systems, and trace semantics)

Other System Models

- Relational semantics
- Labeled transition systems
- State machines
- Probabilistic systems

…can define hyperproperties for all these

Probabilistic Hyperproperties

To incorporate probability:

- Assume probability on state transitions
- Construct probability measure on traces [Halpern 2003]
- Use measure to express hyperproperties

We've expressed:

- **Probabilistic noninterference** [Gray and Syverson 1998]
- Quantitative leakage
- Channel capacity

Probabilistic Noninterference

[Gray 1991, O'Neill et al 2006]

Intuition: the probability of every low trace is the same for every low-equivalent initial state

PR = {T | T is a probabilistic system}

PNI = { T in PR | forall s1,s2 in Init(T),
low(s1) = low(s2) implies (forall finite t,
Pr(s1,T)[{t'|low(t)=low(t')}]
= Pr(s2,T)[{t'|low(t)=low(t')}]) }

Init(T) = set of initial states in T

Pr(s,S) = probability measure on sets of finite traces induced by S from an initial state s

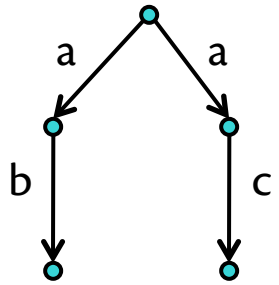
Labeled Transition Systems

To model: encode LTS into trace semantics

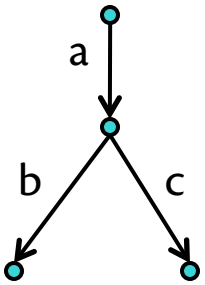
We've expressed **Bisimulation nondeducibility on compositions (BNDC)** [Focardi and Gorrieri FOSAD 2001]

(I'll omit the formalization here)

Encoding LTS's



{ab, ac}

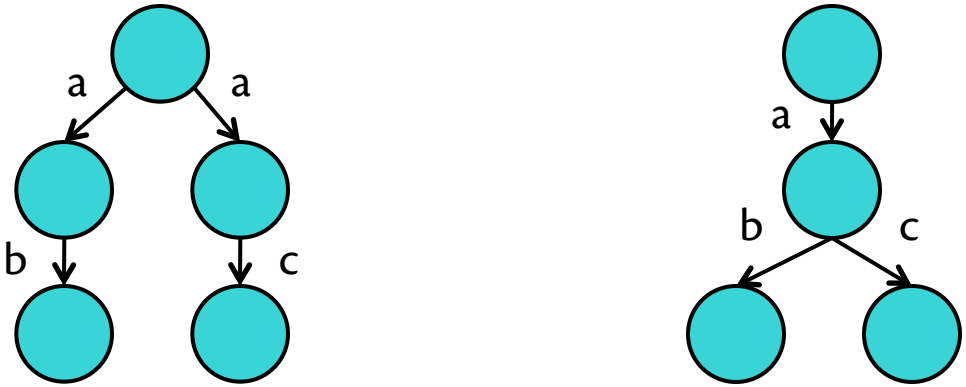


{ab, ac}

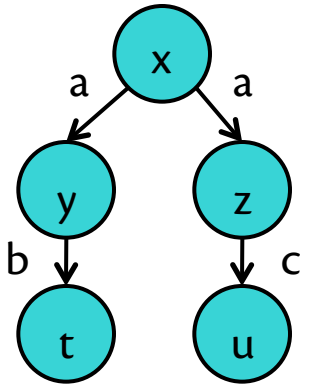
Different LTS's, same trace sets.

But with richer state...

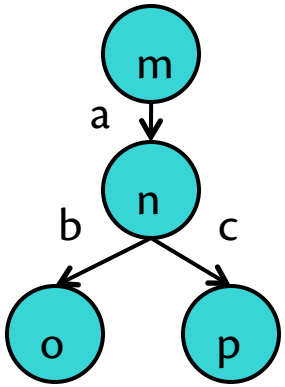
Representing LTS's



Representing LTS's



{(xa)(yb)(t$), (xa)(zc)(u$)}



{(ma)(nb)(o$), (ma)(nc)(p$)}

Encode transition into prior state...

Different LTS's, different trace sets

Other System Models

- Relational semantics
- Labeled transition systems
- State machines
- Probabilistic systems

…can define hyperproperties for all these

Hyperproperties

- Safety and liveness?
- Verification?

Questions??

Safety

Safety proscribes "bad things"

- A bad thing is finitely observable and irremediable
- S is a safety property [Lamport 85] iff

$$(\forall\, t \notin S : (\exists\, b \leq t : (\forall\, u \geq b : u \notin S)))$$

b is a finite trace

$\notin S$

Safety

Safety proscribes "bad things"

- A bad thing is finitely observable and irremediable
- S is a safety property [Lamport 85] iff

$$(\forall\, t \notin S : (\exists\, b \leq t : (\forall\, u \geq b : u \notin S)))$$

b is a finite trace

Safety

Access control is safety:

- Bad thing is a subject completing an operation without the required rights
- Finitely observable: occurs at the moment operation is completed
- Irremediable: once operation is completed, can't "undo" that

Safety

Safety proscribes "bad things"

- A bad thing is finitely observable and irremediable
- S is a safety property [Lamport 85] iff

$$(\forall\, t \notin S : (\exists\, b \leq t : (\forall\, u \geq b : u \notin S)))$$

b is a finite trace

- S is a **safety hyperproperty** ("hypersafety") iff

$$(\forall\, T \notin \boldsymbol{S} : (\exists\, B \leq T : (\forall\, U \geq B : U \notin \boldsymbol{S})))$$

B is a finite set of finite traces

Prefix Ordering

An **observation** is a finite set of finite traces

Intuition: Observer sees a set of partial executions

$M \leq T$ (M is a **prefix** of T) iff:

- M is an observation, and
- $\forall m \in M : (\exists t \in T : m \leq t)$
- If observer watched longer, M could become T

Other definitions are possible...

Powerdomains

We use the *lower (Hoare) powerdomain*

- Our ≤ is the Hoare order

Other powerdomains?

- Change the notion of "observable"
  - Upper: observations can disappear; impossibility of nondeterministic choices becomes observable
  - Convex: similar
- But might be useful on other semantic domains

Hypersafety

Observational determinism

OD = { T | forall t,t' in T, low-in(t[0]) = low-in(t'[0])
implies low(t) = low(t')}

Intuition: bad thing is a pair of traces that cause system to look nondeterministic to low observer

Formally: if T is not in OD, then exists a t,t' in T such that low-in(t[0])=low-in(t'[0]) and exists i such that low(t[..i]) <> low(t'[..i]); no matter how {t[..i],t'[..i]} is extended, cannot get into OD.

Hypersafety

Noninterference

GMNI = { T in SM | forall t in T, exists t' in T,
high-in(t') = ε and low(t) = low(t') }

Intuition: Bad thing is a pair of traces where removing high commands does change low observations

Formally: if T is not in GMNI, then (i) either T not in SM or (ii) exists t in T s.t. for all t' in T, high-in(t') <> ε or exists j such that low(t[..j]) <> low(t'[..j]); that {t[..j]} cannot be extended to get into GMNI.

Hypersafety

Noninterference

GMNI = { T in SM | forall t in T, exists t' in T,
high-in(t') = ε and low(t) = low(t') }

Intuition: Bad thing is a pair of traces where removing high commands does change low observations

Formally: if T is not in GMNI, then (i) either T not in SM or (ii) exists t in T s.t. for all t' in T, high-in(t') <> ε or exists j such that low(t[..j]) <> low(t'[..j]); that {t[..j]} cannot be extended to get into GMNI.

...so not quite hypersafety

Hypersafety for a Rep

System representation: set Rep of trace sets defining a class of systems

e.g., all state machines, all LTSs, all probabilistic systems,

Observation: Obs(Rep) is the all finite sets of finite traces from Rep

Hypersafety for a Rep

S is hypersafety for Rep iff

$$(\forall T \notin \boldsymbol{S} : (\exists B \leq T : (\forall U \geq B : U \notin \boldsymbol{S})))$$

where *T* and *U* must be in Rep, and *B* must be in Obs(Rep)

...GMNI is hypersafety for SM, because T's outside of SM no longer matter

Liveness

Liveness prescribes "good things"

- A good thing is always possible and possibly infinite
- L is a liveness property [AS85] iff

$$(\forall\, t \,:\, (\exists\, g \geq t \,:\, g \in L))$$

t is a finite trace

Liveness

Guaranteed service is liveness:

- Good thing is that every request for service eventually receives a response
- Always possible: a trace in which requests haven't yet been responded to can be extended to include response
- Possibly infinite: in this case, not infinite
  - *Starvation freedom* (progress infinitely often) would be infinite

Liveness

Liveness prescribes "good things"

- A good thing is always possible and possibly infinite
- L is a liveness property [AS85] iff

$$(\forall\, t \,:\, (\exists\, g \geq t \,:\, g \in L))$$

t is a finite trace

- L is a **liveness hyperproperty** ("hyperliveness") iff

$$(\forall\, T \,:\, (\exists\, G \geq T \,:\, G \in \boldsymbol{L}))$$

T is a finite set of finite traces

Hyperliveness

Generalized noninterference

GNI = { T | forall t1,t2 in T, exists t3 in T,
high-in(t3) = high-in(t1) and low(t3) = low(t2) }

Intuition: Good thing is that there exists such a t3

Formally: Given any observation T, add enough additional traces to make sure the resulting system is closed under the existence property required by GNI.

...closure

Liveness Hyperproperties

Possibilistic information flow

- Class of policies requiring "alternate possible explanations" to exist, e.g., GNI, nondeducibility [Sutherland 1986]
- McLean [1996] shows that PIFs can be expressed as closure w.r.t. *selective interleaving functions*
- Mantel [2000] shows that PIFs can be expressed with *closure operators*

Theorem. *All PIF policies are hyperliveness.*

(because any observation can be completed by its closure)

Hyperliveness

Mean response time

RT = { T | mean ({respTimes(t) | t in T}) <= 1 }

Intuition: Good thing is that mean time is low enough

Formally: Given an observation T with any mean response time, synthesize a full system with (if necessary) enough fast responses to drive mean time low enough.

...but what if that kind of synthesis isn't possible for systems of interest?

Hyperliveness for a Rep

L is hyperliveness for Rep iff

$$(\forall T : (\exists G \geq T : G \in \boldsymbol{L}))$$

where *T* must be in Obs(Rep), and *G* must be in Rep

...BNDC turns out to be hyperliveness for LTSs

Relating Properties and Hyperproperties

Can **lift** property T to hyperproperty $[T]$

Satisfaction is equivalent iff $[T]$ = powerset(T)

Theorem. *S is safety implies $[S]$ is hypersafety.*

Theorem. *L is liveness implies $[L]$ is hyperliveness.*

…Verification techniques for safety and liveness carry forward to hyperproperties

Safety and Liveness is a Basis (still)

Theorem. *Every hyperproperty is the intersection of a safety hyperproperty and a liveness hyperproperty.*

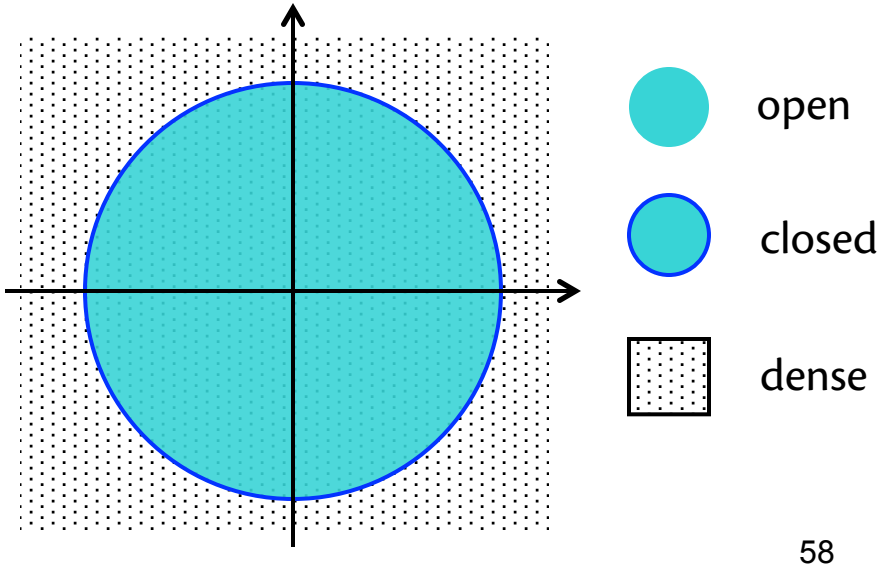
A fundamental basis…

Topology

Open set: Can always "wiggle" from point and stay in set

Closed set: "Wiggle" might move outside set

Dense set: Can always "wiggle" to get into set



Topology of Hyperproperties

For **Plotkin topology** on properties [AS85]:

- Safety = closed sets
- Liveness = dense sets

We can define a topology O in which...

Theorem. *Hypersafety = closed sets of O.*

Theorem. *Hyperliveness = dense sets of O.*

Theorem. *Our topology O is equivalent to the* lower Vietoris construction *applied to the Plotkin topology.*

Stepping Back…

- Safety and liveness? ✓
- Verification?

Verification of properties

- Partly-automated verification for any property, based on *safety* and *liveness*
  [Alpern and Schneider 1987; etc.]
- Automated verification for classes of properties, based on *model checking*
  [Clarke, Emerson, Sistla 1986; etc.]
- Manual verification for classes of properties, based on *logical proof systems*
  [Gabay et al. 1980; etc.]

Formalize and verify any property? 

Logic and Verification

Temporal logic: LTL, CTL*?

- Highly successful for trace properties
- But not for security policies [McLean 1994, Alur et al. 2006]
- Why not? Only a single trace in scope...

Why not CTL*?

$s \models AA\,\phi$

= for all $\pi \in M$, if $t[0] = s$ then $\pi \models A\,f$

= for all $\pi,\rho \in M$, if $t[0] = u[0] = s$ then $\rho \models \phi$

…only the last trace is "remembered" by semantics

Syntax

LTL: [Pnueli 1977]

$\phi ::= \neg\phi \mid \phi_1 \vee \phi_2 \mid \ldots \mid X\,\phi \mid \phi_1\, U\, \phi_2 \mid \ldots \mid G\,\phi \mid \ldots$

Propositions: x-equals-42

HyperLTL: [Clarkson et al. 2014]

$\psi ::= \forall\, t{:}\,\psi \mid \exists\, t{:}\,\psi \mid \phi$

First-order relations: x-equals-42(t)

…LTL is a fragment of HyperLTL

Example HyperLTL Formula

Observational determinism:

$$\forall t: \forall u: t[0] =_L u[0] \Rightarrow t =_L u$$

$t[0] =_L u[0]$ is syntactic sugar for $\bigwedge_{p \in L} p(t) \Leftrightarrow p(u)$
(first state in both traces agrees on all relations in L)

$t =_L u$ is syntactic sugar for $G\,(t[0] =_L u[0])$
(both traces agrees on all relations in L)

Note: multiple traces in scope; syntax that reads like the "normal" math written in noninteference papers.

Semantics

LTL:

- formula modeled by single trace: $t \models \phi$
- system modeled by set T of traces

HyperLTL:

- formula modeled by **set** of traces *(actually, set of named traces i.e. valuation or environment)*
- system still modeled by set T of traces, which is what quantifiers range over:

$$\Pi \models \forall t : \psi \text{ iff for all } \tau \text{ in T, have } \Pi, t{=}\tau \models \psi$$

Semantics

$\Pi \models \forall t : \psi$ iff for all τ in T, have $\Pi, t{=}\tau \models \psi$

$\Pi \models \exists t : \psi$ iff exists τ in T, s.t. $\Pi, t{=}\tau \models \psi$

$\Pi \models p(t)$ iff $p \in \Pi(t)[0]$

$\Pi \models \neg\phi$ iff $\Pi \models \phi$ doesn't hold

$\Pi \models \phi_1 \vee \phi_2$ iff $\Pi \models \phi_1$ or $\Pi \models \phi_2$

$\Pi \models X\,\phi$ iff $\Pi[1..] \models \phi$

$\Pi \models \phi_1 \cup \phi_2$ iff there exists $i \geq 0$ s.t. $\Pi[i..] \models \phi_2$ and

for all j where $0 \leq j < i$, have $\Pi[j..] \models \phi_1$

Model Checking

- Adapts LTL algorithm based on Büchi automata
  [Wolper et al. 1983, Lichtenstein and Pnueli 1985, Vardi and Wolper 1994, …]
- Prototype implementation…
  - builds automata using self-composition [Barthe et al. 2004],
  - then outsources to GOAL [Tsay et al. 2007] for automata constructions
- Supports fragment of HyperLTL
  - Up to one quantifier alternation, e.g. AE, AAE, EA
  - Suffices for all our information-flow examples
- Yields verification methodology for any *linear-time* hyperproperty

Model Checking: Complexity

Depends on depth of quantifier alternation:

- 0 alternations: $\forall t : \forall u : \phi$
  - e.g., OD, GMNI
  - PSPACE in size of system, NLOGSPACE in size of formula
- 1 alternation:: $\forall t : \exists u : \phi$
  - e.g., noninference, GNI
  - EXPSPACE in size of system, PSPACE in size of formula

Model Checking: Complexity

Depends on depth of quantifier alternation:

- 2 alternations: no information flow property we've studied needs it
- Full HyperLTL extended with arbitrary quantifier alternation (HyperCTL*):
  - Decidable. Elementary. ☺

Other Related Logics

- SecLTL [Dimitrova et al. 12]
  - LTL plus *hide* modality (high hidden from observation)
  - Designed for output-deterministic systems
    - Can't seem to express, e.g., generalized noninteference
- L_{KU} [Balliu et al. 11]
  - Linear-time logic of knowledge
  - Handles declassification policies
  - Designed for observational determinism
    - Can't seem to express, e.g., noninference
- Incremental hyperproperties [Milushev and Clarke 12]
  - *Polyadic* modal mu-calculus [Andersen 94]
    - Models are tuples of transition systems
  - Verifiable by *game-based* model checking
  - HyperLTL is simpler and seems to suffice for any state-based information-flow policy

Stepping Back…

- Safety and liveness? ✓
- Verification?
  - Model-checking (expensive) ✓
  - Reduce to trace properties
  - Refinement

Verification of 2-Safety

2-safety: "Property that can be refuted by observing two finite traces" [Terauchi and Aiken 2005]

Methodology:

- Transform system with **self-composition construction** [Barthe, D'Argenio, and Rezk 2004]
- Verify safety property of transformed system
  - Implies 2-safety property of original system

…Reduction from hyperproperty to property

k-Safety Hyperproperties

A ***k*-safety hyperproperty** is a safety hyperproperty in which the bad thing never has more than *k* traces

$$(\forall T \notin \boldsymbol{S} : (\exists B \leq T : |B| \leq k \ \wedge \ (\forall U \geq B : B \notin \boldsymbol{S})))$$

Examples:

- **1-hypersafety:** the lifted safety properties
- **2-hypersafety:** Terauchi and Aiken's 2-safety properties
- ***k*-hypersafety:** $SEC(k)$ = "System can't, across all runs, output all shares of a *k*-secret sharing"
- **Not *k*-hypersafety for any *k*:** $SEC = \bigcup_k SEC(k)$

Verifying *k*-Hypersafety

Theorem. *Any k-safety hyperproperty of* S *is equivalent to a safety property of* S^k.

➔ Yields methodology for *k*-hypersafety

Refinement Revisited

Stepwise refinement:

- Development methodology for properties
  - Start with specification and high-level (abstract) program
  - Repeatedly **refine** program to lower-level (concrete) program
- Techniques for refinement well-developed

Long-known those techniques don't work for security policies—i.e., hyperproperties

- Develop new techniques?
- Reuse known techniques?

Refinement Revisited

Theorem. *Known techniques work with all hyperproperties that are* subset-closed.

Theorem. *All safety hyperproperties are subset-closed.*

➔ Stepwise refinement applicable with hypersafety

Stepping Back…

- Safety and liveness? ✓
- Verification?
  - Model-checking (expensive) ✓
  - Reduce to trace properties (*k*-safety) ✓
  - Refinement (hypersafety) ✓
  - Logical proof system? (in progress)

…verify by decomposing to safety+liveness?

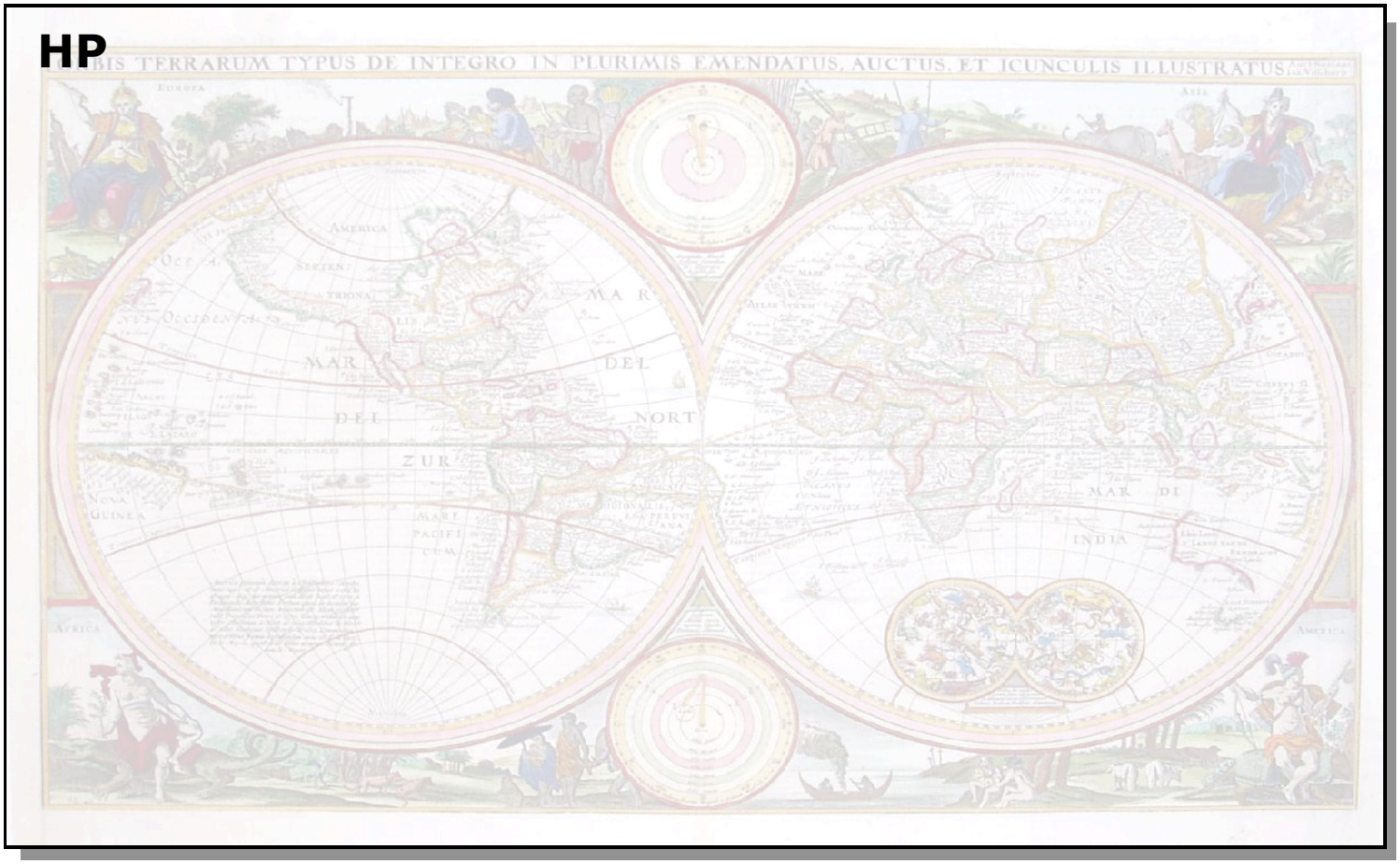
Summary

Theory of hyperproperties :

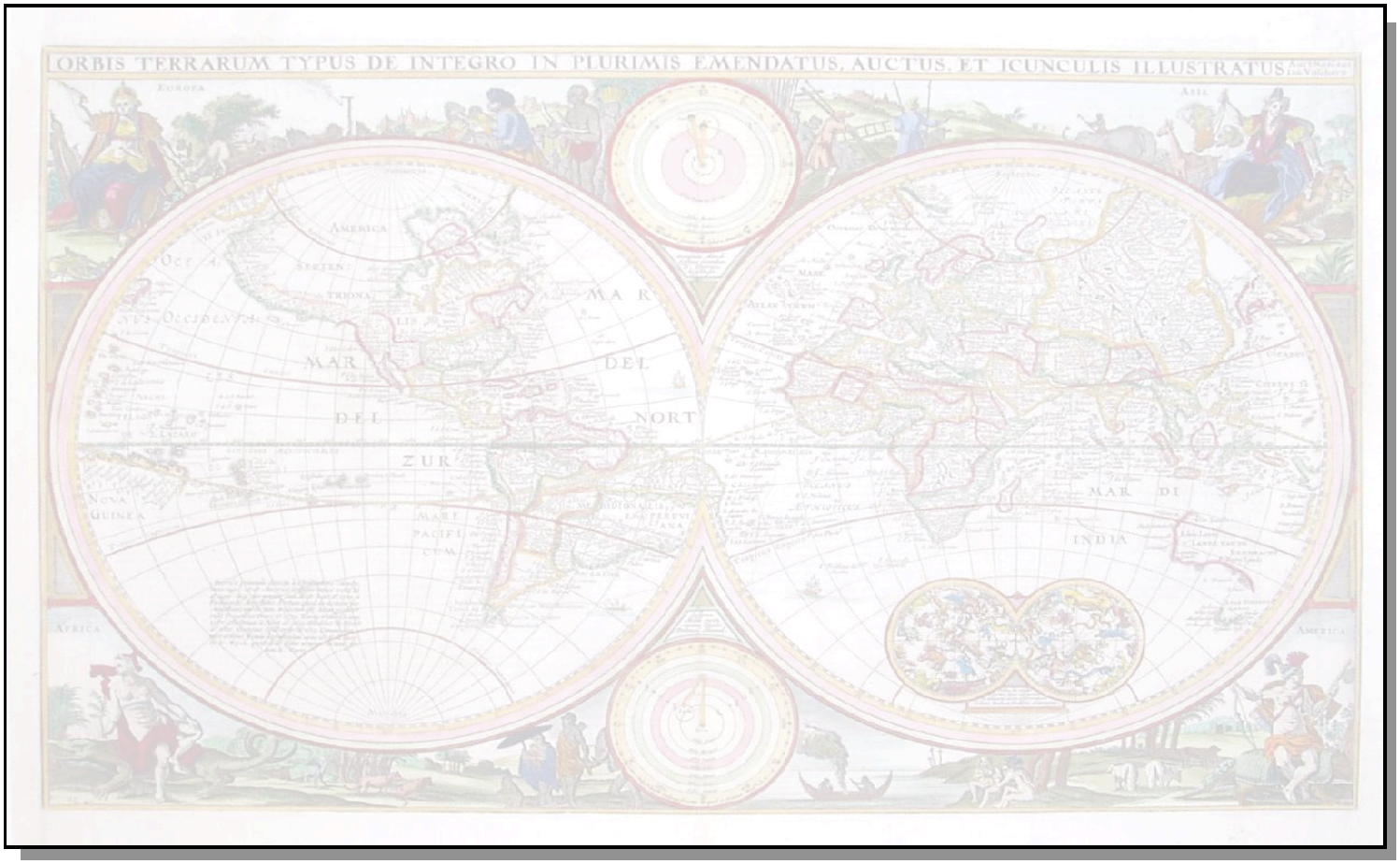
- Parallels theory of properties
  - Safety, liveness (basis, topological characterization)
  - Verification (HyperLTL, *k*-hypersafety, stepwise refinement)
- Expressive completeness
- Enables classification of security policies…

Charting the landscape…

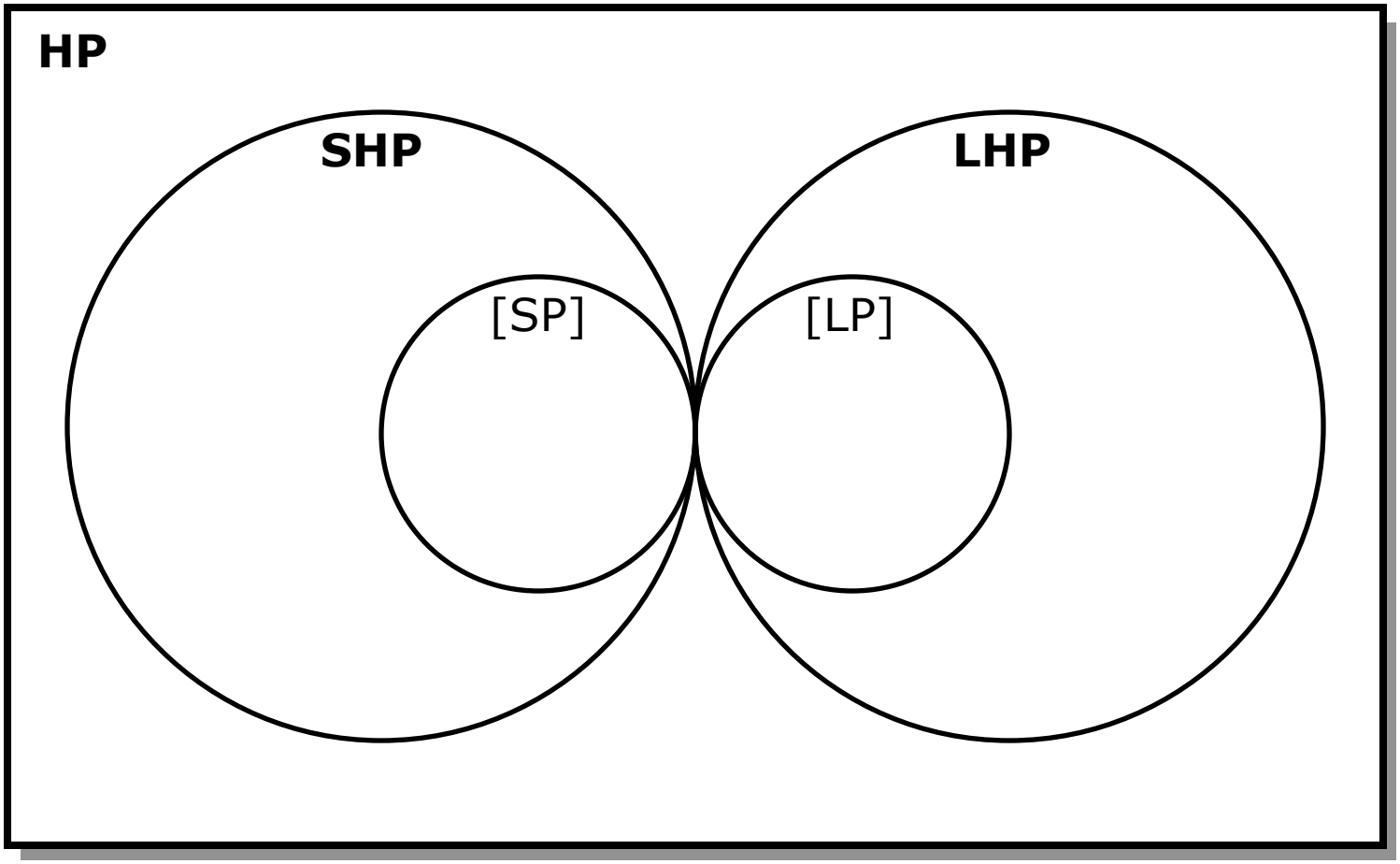
HP



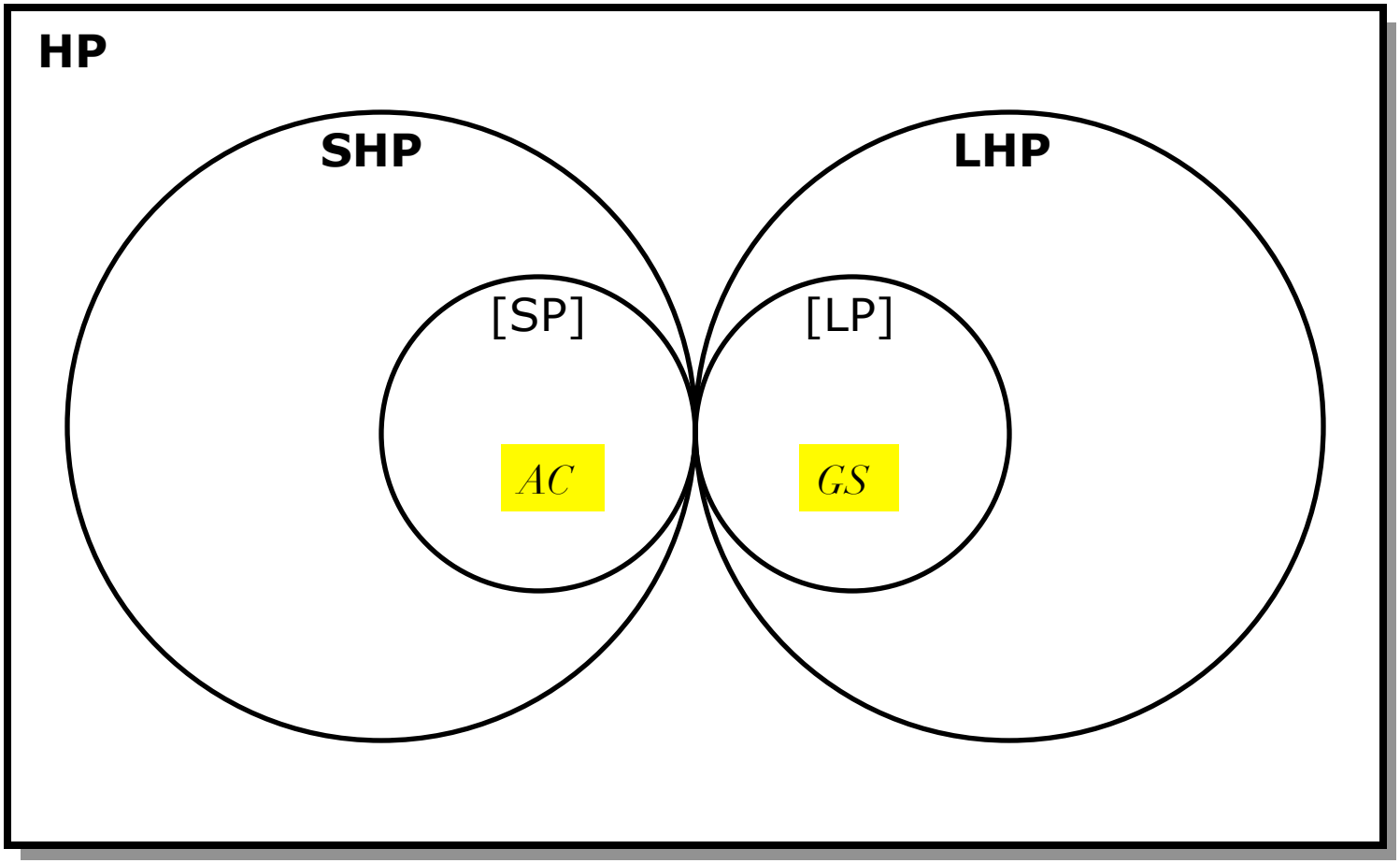
All hyperproperties (**HP**)



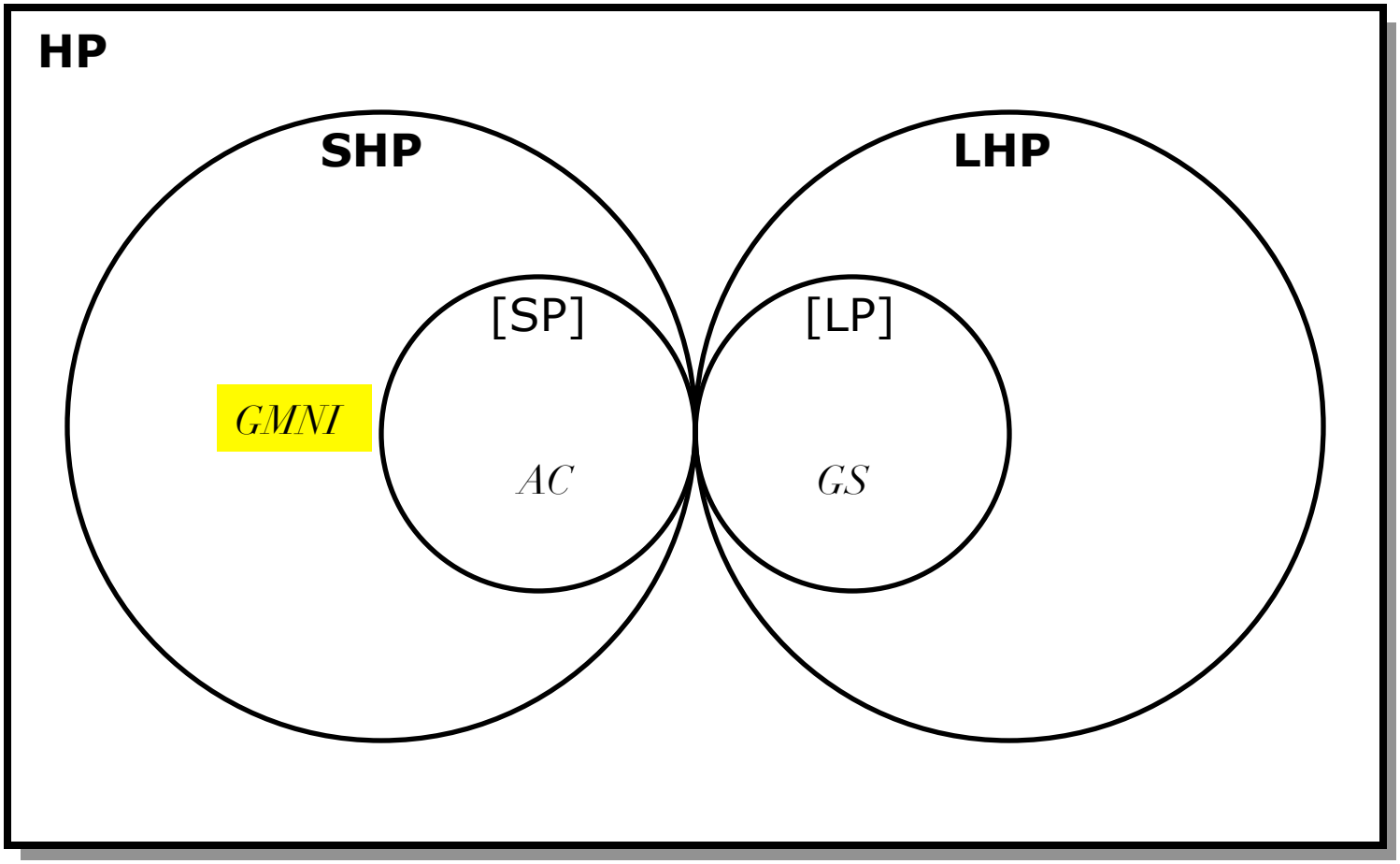
Safety hyperproperties (**SHP**)
Liveness hyperproperties (**LHP**)



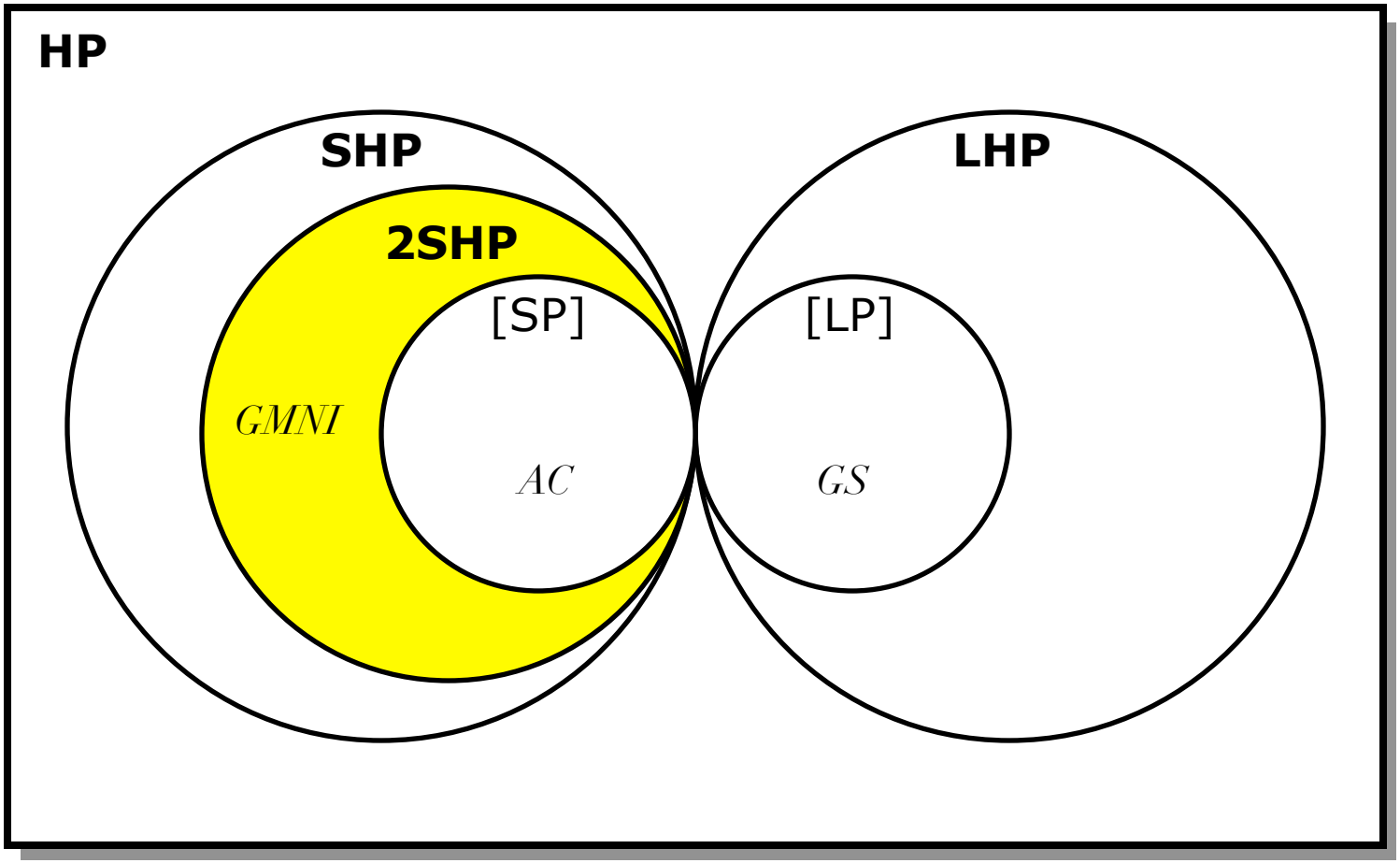
Lifted safety properties [SP]
Lifted liveness properties [LP]



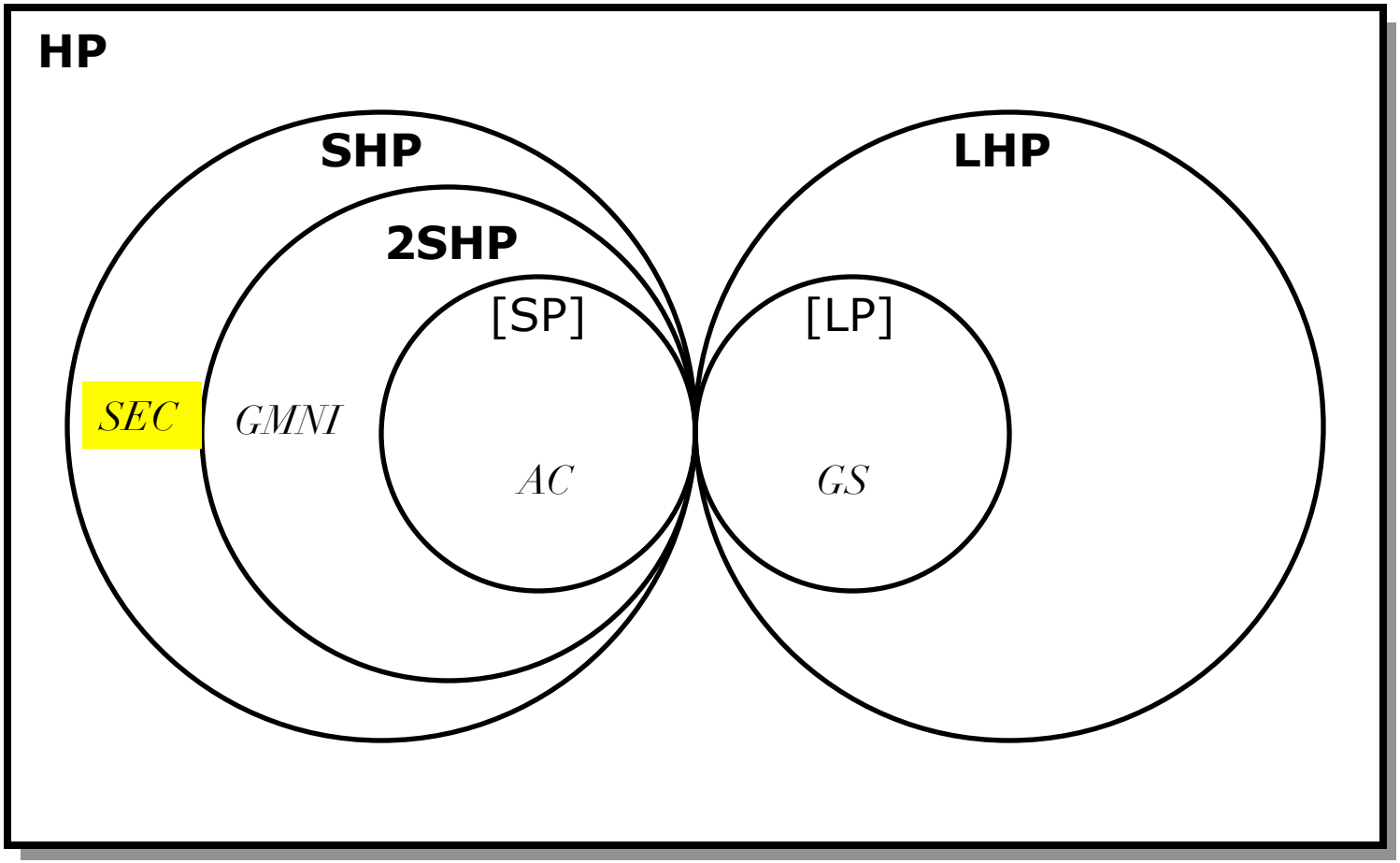
Access control (AC) is safety
Guaranteed service (GS) is liveness



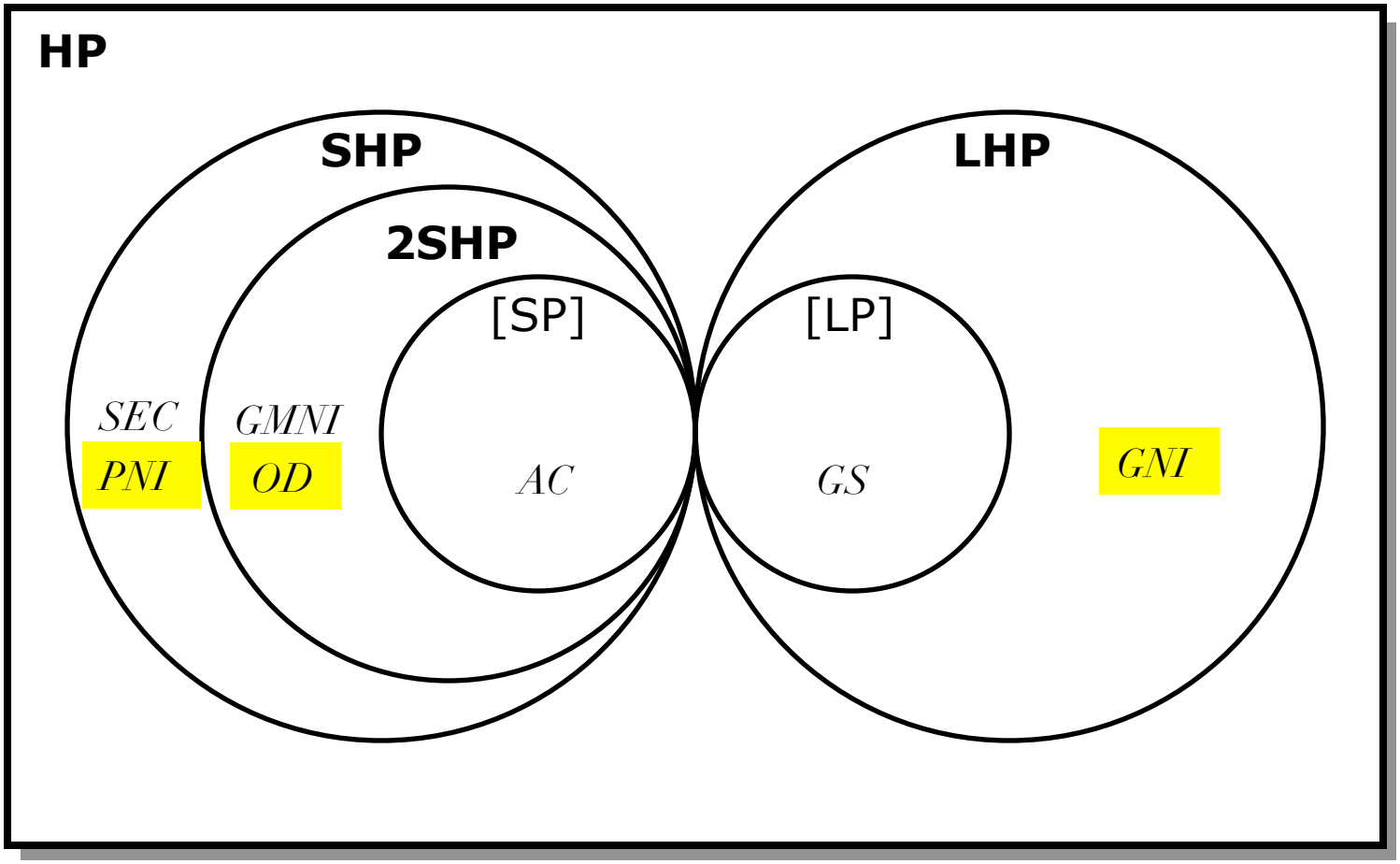
Goguen and Meseguer's noninterference (*GMNI*) is hypersafety



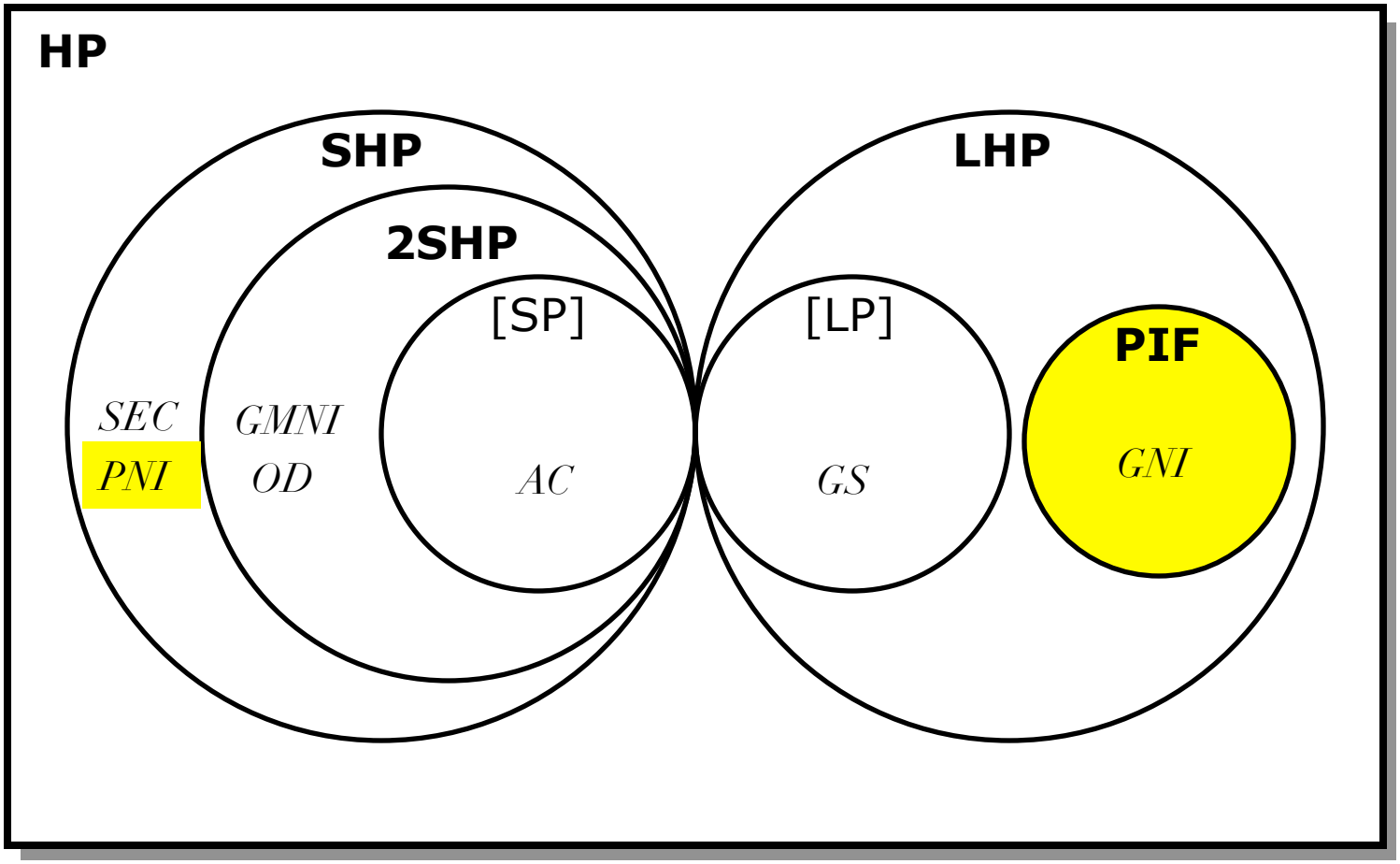
2-safety hyperproperties (**2SHP**)



Secret sharing (SEC) is not k-hypersafety for any k



Observational determinism (OD) is 2-hypersafety
Generalized noninterference (GNI) is hyperliveness
Probabilistic noninterference (PNI) is hypersafety



Possibilistic information flow (**PIF**) is hyperliveness

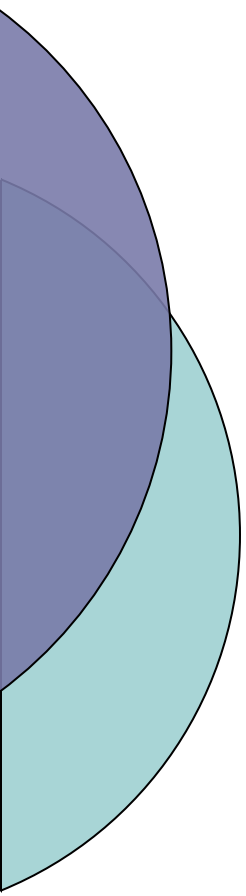
Revisiting the CIA Landscape

- **Confidentiality**
  - Information flow is not a property
  - Is a hyperproperty (HS: OD; HL: GNI)
- **Integrity**
  - Safety property?
  - Dual to confidentiality, thus hyperproperty?
- **Availability**
  - Sometimes a property (max. response time)
  - Sometimes a hyperproperty (HS: % uptime, HL: RT)

➔ CIA seems unrelated to hyperproperties

Reading

- **Hyperproperties.** *Journal of Computer Security* 18(6): 1157–1210, 2010. With Fred B. Schneider.
- **Temporal Logics for Hyperproperties.** In *Proc. POST 2014*, p. 265–284. With Bernd Finkbeiner, Masoud Koleini, Kristopher Micinski, Markus Rabe, and Cesar Sanchez.



Hyperproperties

Michael Clarkson
Cornell University

15th International School on Foundations of Security Analysis and Design
University Residential Center of Bertinoro, Italy
September 3, 2015