A Petri Net View of Mobility

(FORTE 2005)

Charles Lakos

Roadmap

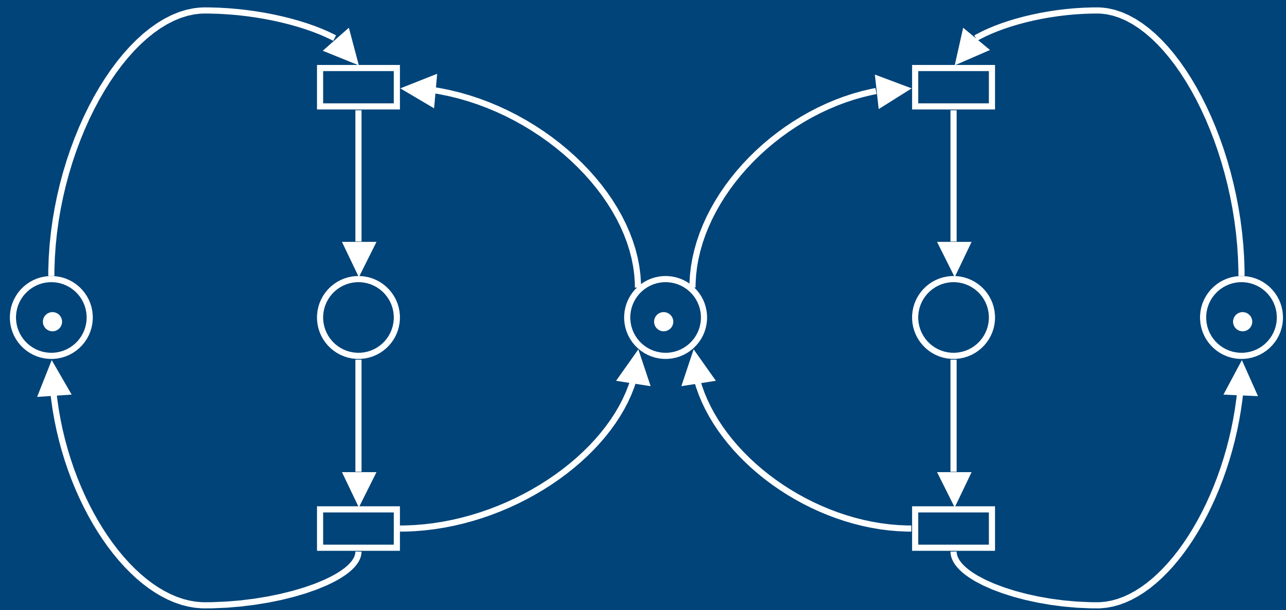
- Introduction/reminder of Petri Net formalism
- Key issues of mobility
- Previous approach – the Hamburg group

- The proposal – based on modular nets
- Coloured version and the notion of garbage

- Conclusions

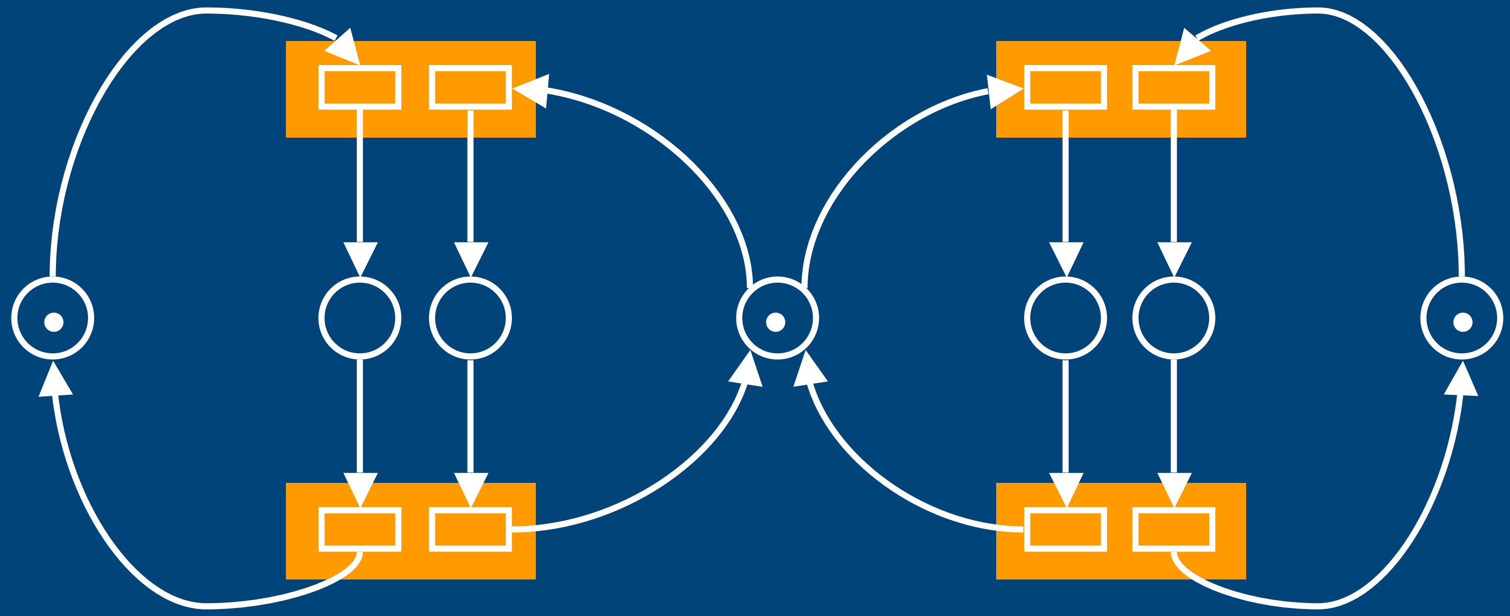
Petri Nets

- Net structure – places, transitions, arcs
- System behaviour – markings, steps



Petri Nets

- Modular structure – place fusion, transition fusion



Mobile systems

- Expose the interplay between *locality* and *connectivity* (Milner)
- ***Connectivity*** involves having a reference and being able to dereference it
- ***Locality*** constrains what you can dereference
- A simple and general Petri Net solution has proved elusive

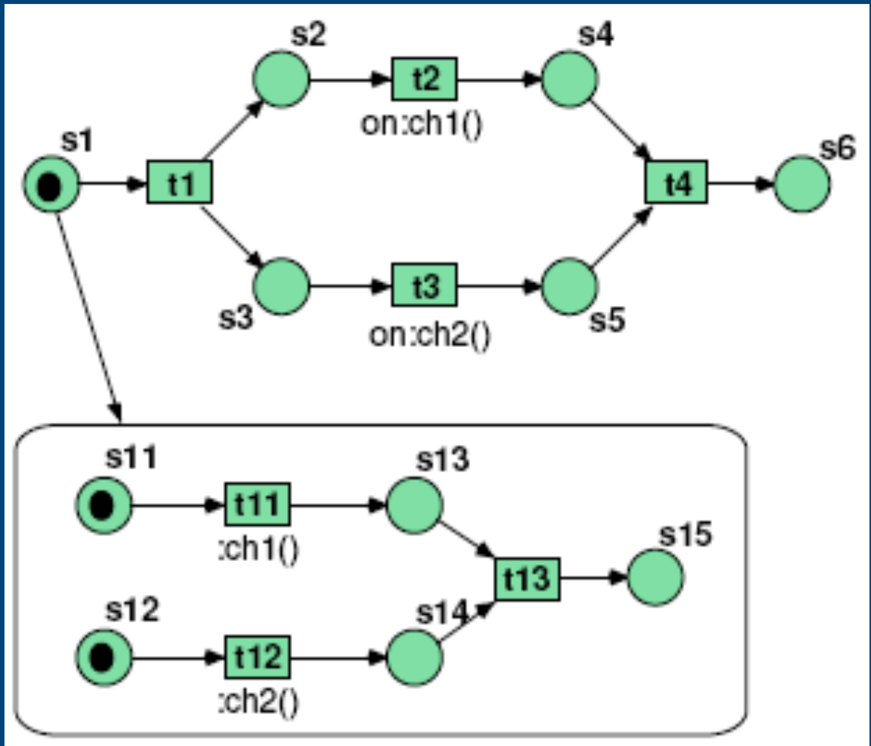
Nets-within-nets paradigm (Hamburg)

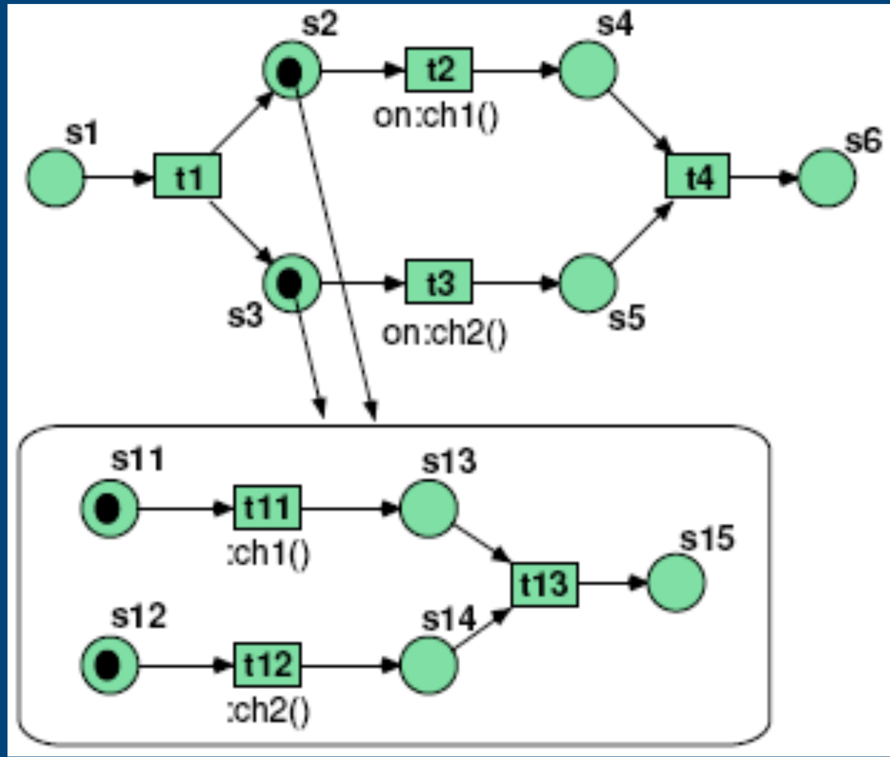
- (At least) two levels of nets:

- ***System net*** has tokens which are black tokens or object nets
- ***Object nets*** have black tokens

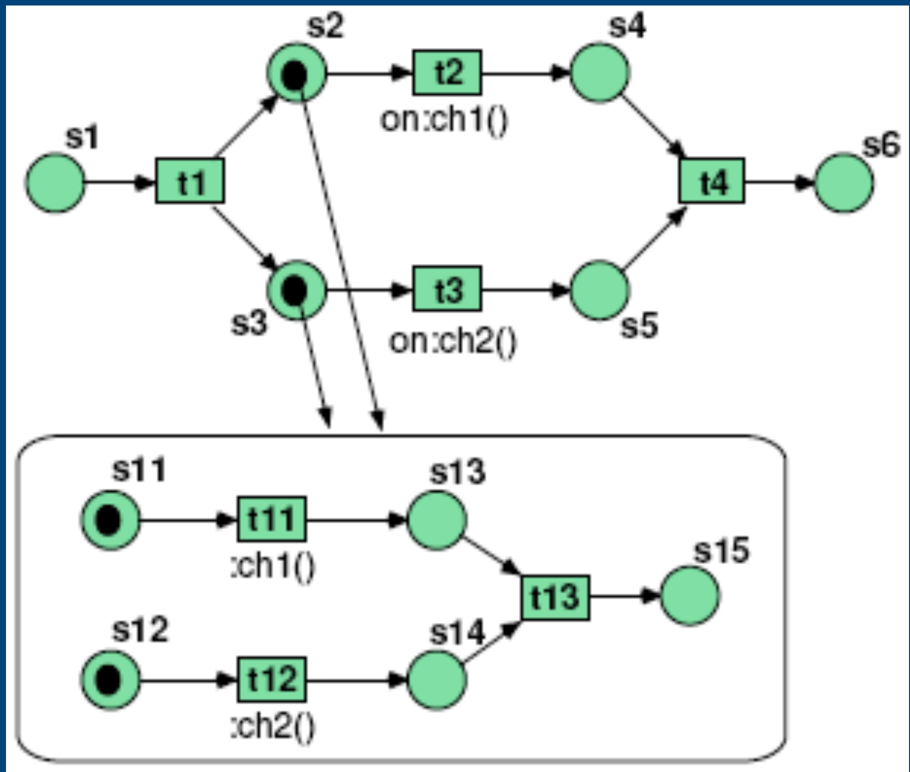
- ***Reference semantics*** – tokens can be Object net references
- ***Value semantics*** – tokens can be Object net instances
- ***History process semantics*** – tokens can be Object net processes

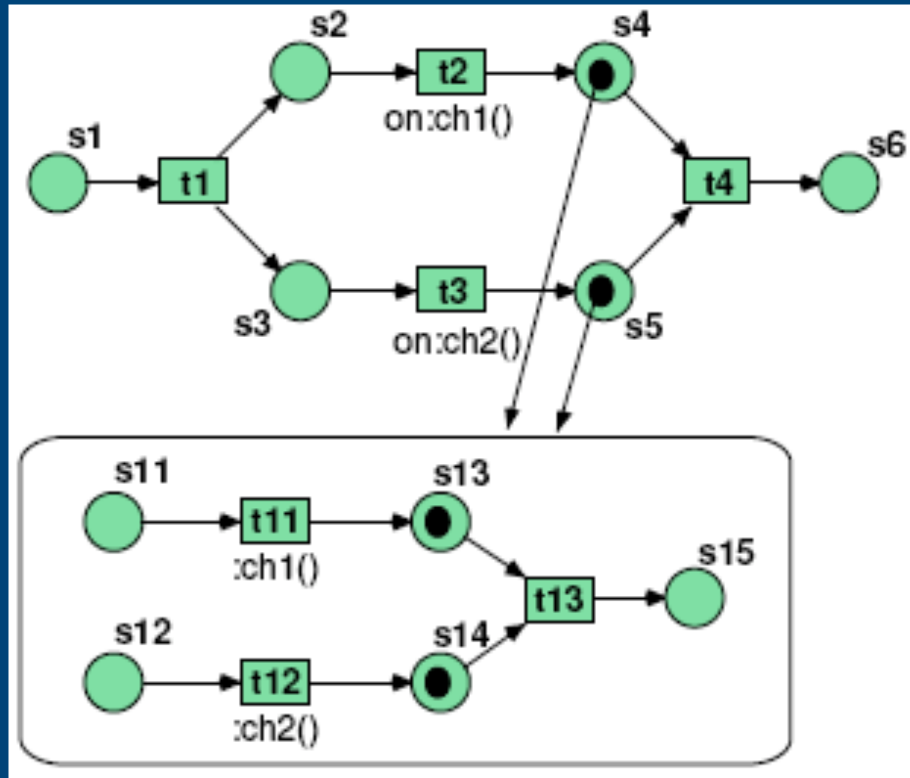
Nets-within-nets – reference semantics



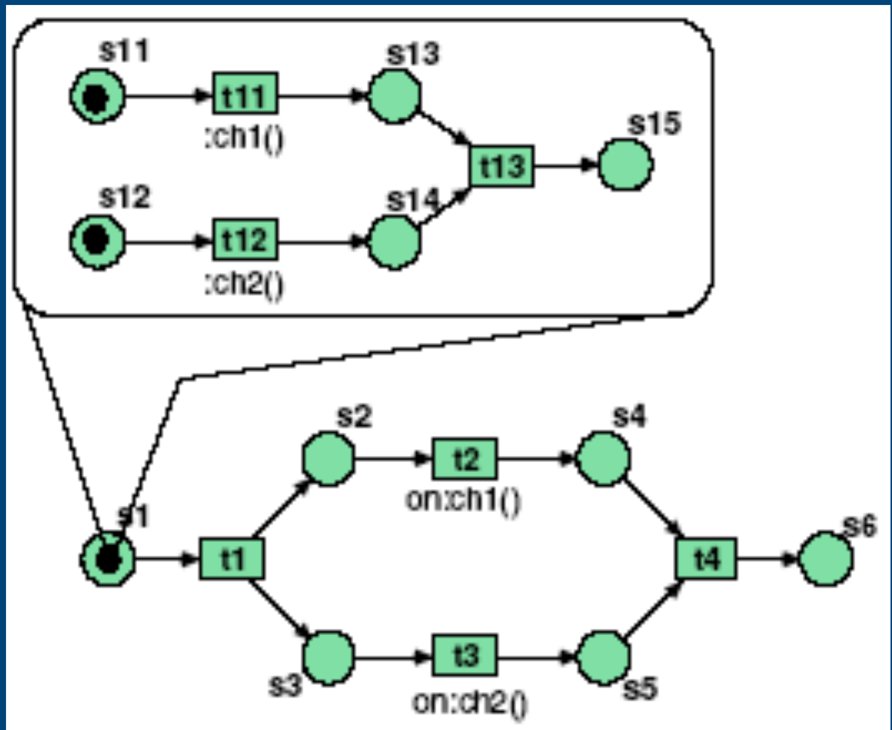


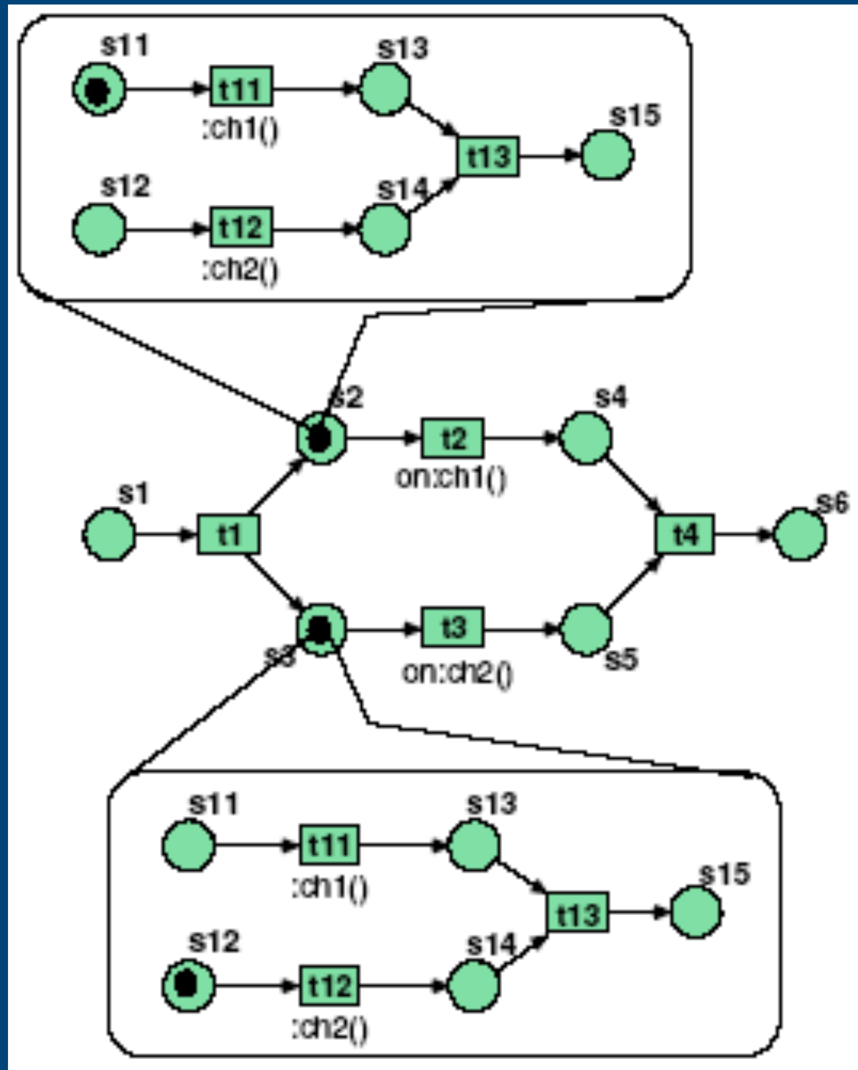
Nets-within-nets – reference semantics



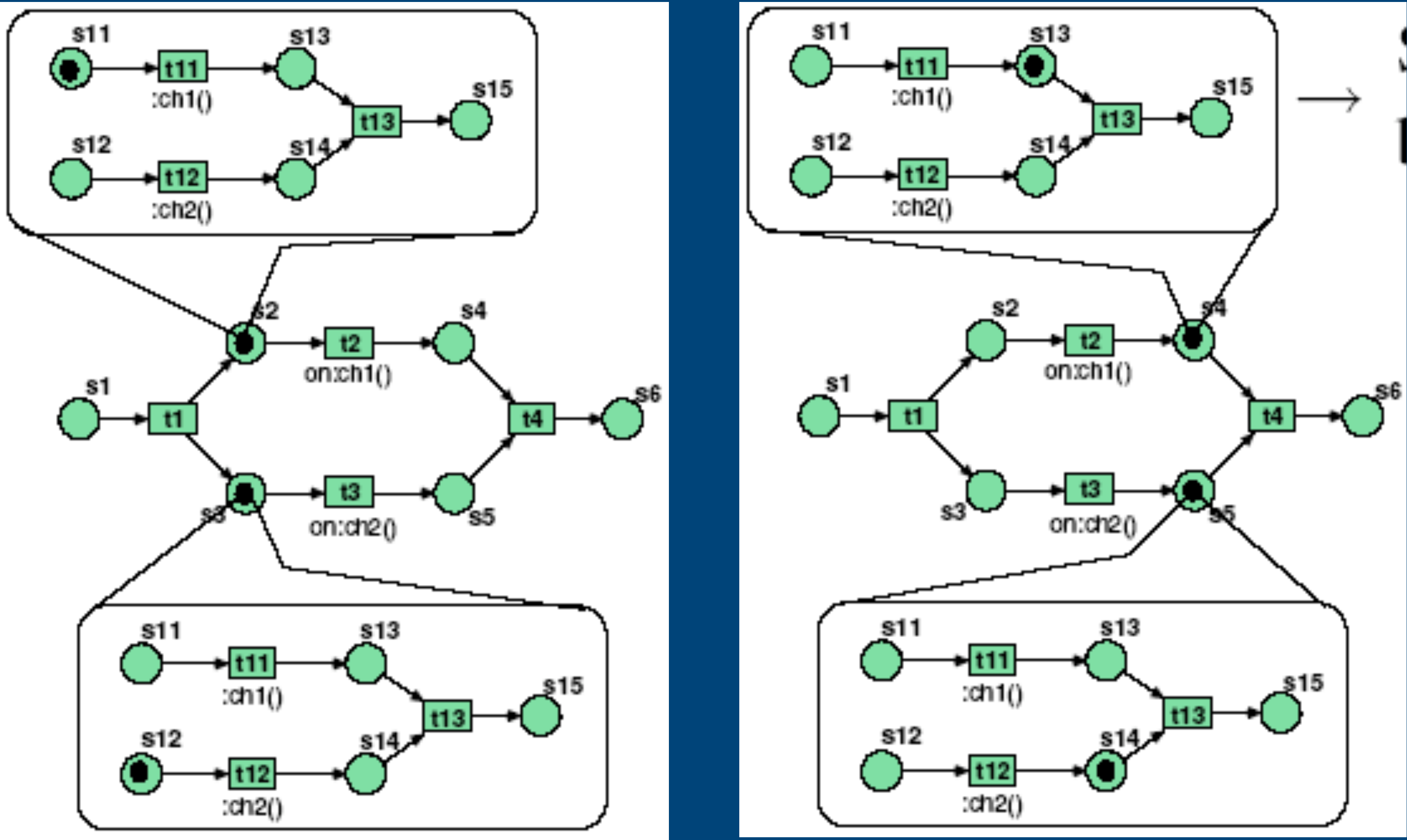


Nets-within-nets – value semantics





Nets-within-nets – value semantics



Nets-within-nets Limitations

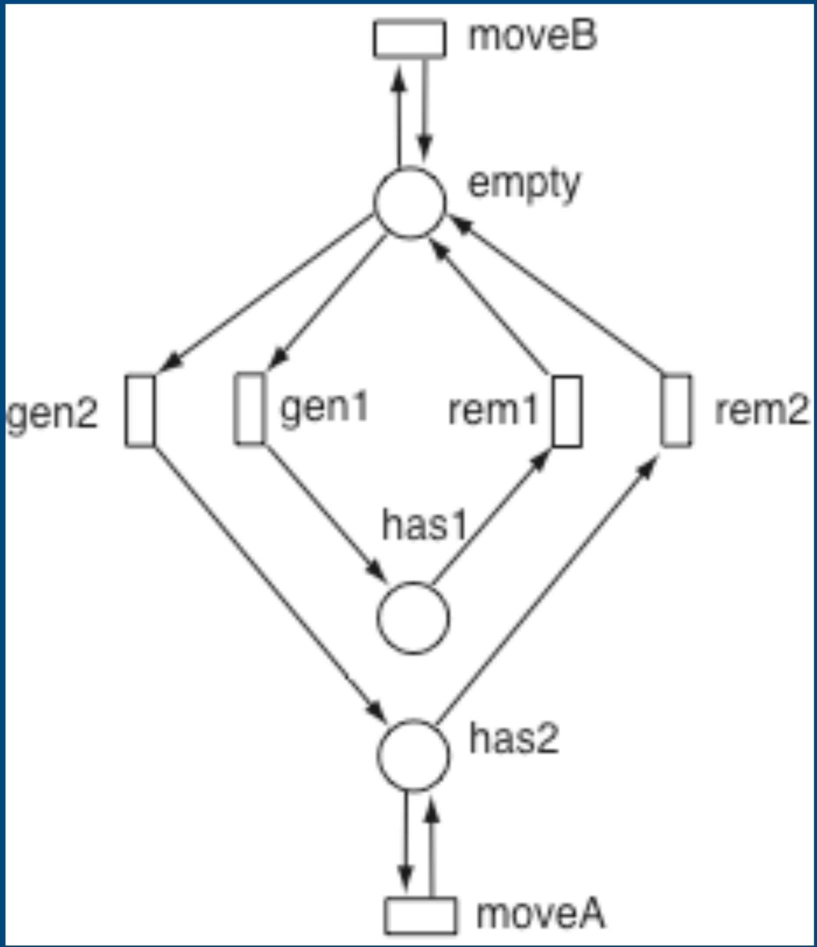
- Either ***value*** or ***reference*** or … semantics
  - Value semantics gives notion of locality
  - Reference semantics gives notion of connectivity
- Limited interaction
  - object net can only interact with transitions adjacent to place
- Formal results are for very limited examples
  - One system net and one (instance of an) object net
  - Value semantics is more powerful than reference semantics
- Examples with ***Renew*** are not very persuasive

Proposal for mobile nets

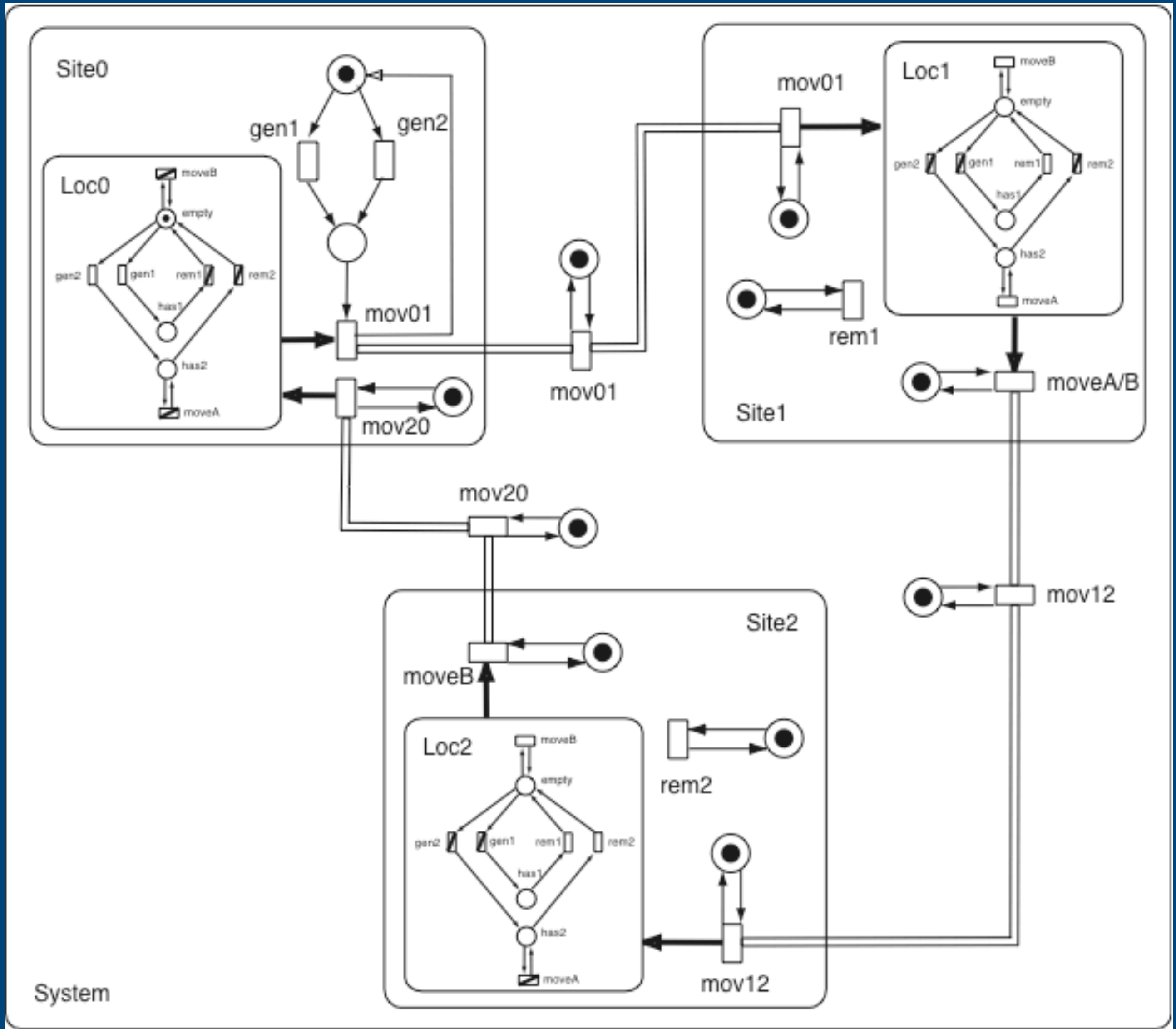
- Start with ***modular nets***
  - have a number of Petri Nets – called ***modules*** or ***subnets***
  - combined by place and transition fusion

- Extend the distinction between a ***net*** and a ***system*** …
  - ***Subnet*** captures the structure of a module
  - ***Location*** = subnet + fusion context
  - ***Subsystem*** = location with a non-empty marking

Mail agent – a subnet



Mail system



Subnets
Locations
Subsystems

Fusions
Shifting locations

Nets and locations

- Nets (and subnets) are standard

Definition 2 (Petri Net). *A Petri Net (PN) is a tuple $PN = (P, T, W)$ where:*

1. *P is a finite set of places.*
2. *T is a finite set of transitions with $P \cap T = \emptyset$.*
3. *$W : (P \times T) \cup (T \times P) \to \mathbb{N}$ is an arc weight function.*

- Locations can be nested (and have a fusion context)

Definition 5 (PN Location). *A Petri Net Location is a tuple $L = (S_L, P_L, T_L, W_L)$ where:*

1. *S_L is a finite set of locations. We define $loc(L) = \bigcup_{s \in S_L} loc(s) \cup \{L\}$. We require $\forall s \in S_L : loc(s) \cap \{L\} = \emptyset$.*
2. *(P_L, T_L, W_L) is a Petri Net. We define $plc(L) = \bigcup_{s \in S_L} plc(s) \cup \{P_L\}$ and $trn(L) = \bigcup_{s \in S_L} trn(s) \cup \{T_L\}$.*

Mobile systems

- Convenient to specify fusion at the level of the system
  - for convenience we assume transitive closure of place fusion sets
  - for convenience we require consistency of transition fusion sets

Definition 6 (Mobile System). *A Mobile System is a tuple* $MS = (L_0, PF, TF, M_0)$ *where:*

1. L_0 *is a location, called the* root location. *We define* $P = plc(L_0)$ *and* $T = trn(L_0)$.
2. PF *is a set of place fusion sets where* $\bigcup_{pf \in PF} pf = P$ *and* $\forall pf_1, pf_2 \in PF$: $pf_1 \cap pf_2 \neq \emptyset \Rightarrow pf_1 = pf_2$.
3. TF *is a set of transition fusion sets where* $\bigcup_{tf \in TF} tf = T$ *and* $\forall tf_1, tf_2 \in TF$: $tf_1 \cap tf_2 \neq \emptyset \Rightarrow |tf_1| = |tf_2|$.
4. M_0 *is the initial marking of the location.*

Classify places and transitions

- ***Local*** vs ***exported*** – determined by size of fusion sets
- ***Vacate*** vs ***occupy*** vs ***regular*** – determined by arcs incident on local places

Definition 9. *For a Mobile System MS we classify places and transitions as follows:*

1. $LP = \{p \in P \mid \exists pf \in PF : pf = \{p\}\}$ *is the set of* local places.
2. $EP = P - LP$ *is the set of* exported places.
3. $LT = \{t \in T \mid \exists tf \in TF : tf = \{t\}\}$ *is the set of* local transitions.
4. $ET = T - LT$ *is the set of* exported transitions.
5. $VT = \{t \in T \mid \exists p \in LP : W(p,t) > 0 \wedge \forall p \in P : W(t,p) = 0\}$ *is the set of* vacate transitions.
6. $OT = \{t \in T \mid \exists p \in LP : W(t,p) > 0 \wedge \forall p \in P : W(p,t) = 0\}$ *is the set of* occupy transitions.
7. $RT = \{t \in T \mid \exists p_1, p_2 \in LP : W(t,p_1) > 0 \wedge W(p_2,t) > 0\}$ *is the set of* regular transitions.

Well-formed mobile system

- Need to know whether locations are occupied
- Classification of transitions as ***vacate***, ***occupy***, ***regular*** is consistent and covers all transitions

Definition 10 (Well-formed). *A Mobile System MS is* well-formed *if:*

1. *All transitions are vacate, occupy or regular transitions, i.e.* $T = VT \cup OT \cup RT$.
2. *Vacate transitions empty a location for all reachable markings, i.e.* $\forall L \in loc(L_0) : \forall t \in VT \cap T_L : \forall M \in [M_0\rangle : M[t\rangle M' \Rightarrow \forall p \in LP \cap plc(L) : M'(p) = \emptyset$.
3. *Occupy transitions fill a location for all reachable markings, i.e.* $\forall L \in loc(L_0) : \forall t \in OT \cap T_L : \forall M \in [M_0\rangle : M[t\rangle M' \Rightarrow \forall p \in LP \cap plc(L) : M(p) = \emptyset$.

Isolated subsystem

- An ***isolated subsystem*** has no effect (directly or indirectly) on the root location
  - it can be ignored for the purposes of reachability analysis

Definition 11 (Isolated subsystem). *Given a Mobile System MS in marking M, a transition sequence* $t_1 t_2 ... t_n$ *is a* causal sequence *if there are markings* $M_1, M_2, ... M_n$ *such that* $M[t_1\rangle M_1[t_2\rangle M_2 ... [t_n\rangle M_n$ *and* $\forall k \in 1..(n-1) : \exists p \in P : W(t_k, p) > 0 \wedge W(p, t_{k+1}) > 0$. *Given a Mobile System MS, a subsystem resident in location L is* isolated *in marking M if there is no causal sequence* $t_1 t_2 ... t_n$ *with* $t_1 \in T_L$ *and* $t_n \in T_{L_0}$.

Coloured mobile systems

- Adopt the common approach of using colour to distinguish folded components – see def 20
- Require such colours to be used ***consistently***
  - identifiers determine the associated subsystems
  - distinct subsystems have distinct identifiers
  - tokens in fused places indicate ***all*** subsystems to which it belongs
  - firing modes of fused transitions indicate ***all*** participating subsystems
  - transition firing modes must have identifier in common with tokens
  - transitions cannot ***invent*** identifiers matching existing subsystems

Coloured mobile systems

- Colour makes things more concise but more messy formally
  - there are many choices for coding token and mode colours
  - the imposed regularity helps in determining properties

- Isolated subsystems can be defined using causal sequences
- Can be approximated with garbage collection algorithms
  - for a location to modify the root location, you need fusion
  - fusion requires that the context knows the identifier of the subsystem
  - this condition is sufficient to imply that the subsystem is isolated but it is not a necessary condition

Conclusions

- A natural way to capture mobility in the Petri Net formalism
  - start with modular nets (with general fusion possibilities)
  - differentiate subnets, locations, subsystems
- Well-formed property is based on classifying transitions
  - tells us when a subsystem migrates from one location to another
- Isolated subsystems (garbage) cannot affect the root location
  - notion is difficult to compute precisely – colour helps to approximate
- State space exploration is possible using symmetry techniques
  - see ATVA 2005 paper

State Space Exploration of Object-Based Systems using Equivalence Reduction and the Sweepline Method

(ATVA 2005)

Charles Lakos

Roadmap

- Characteristics of Object-Based Systems
- State space exploration requirements

- Equivalence reduction + Sweepline
- Experimental results

- Conclusions

Object-Based Systems

- Object-Based Systems have the notion of ***objects***
- Object-Oriented Systems also include the notion of ***inheritance***

- "An ***object*** has ***state***, ***behaviour*** and ***identity***"
  - ***state*** = static properties together with their current values
  - ***behaviour*** = how an object acts and reacts with changes of state
  - ***identity*** = property that distinguishes one object from all others

Object Identity

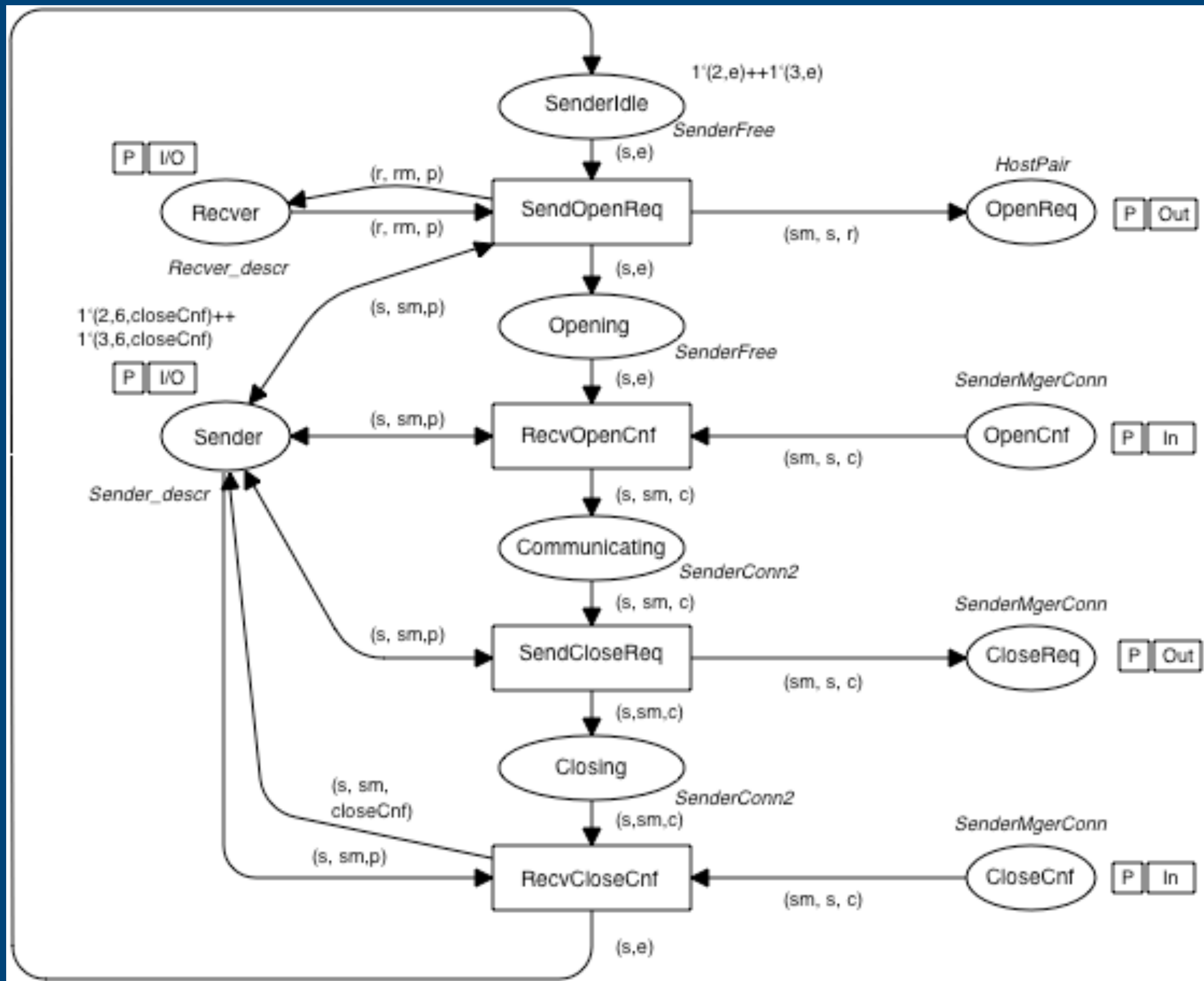
- Object identity is a key feature of object systems
  - it implies some form of reference semantics
- The analysis of object-based systems will require techniques to handle object identity
  - specific object identifiers are not important but only equality
  - allocation of objects in a concurrent/distributed system will result in objects with different identifiers but the same essential configuration
- We need some form of equivalence reduction or graph isomorphism

Range of applicability

- Clearly relevant to the analysis of object-oriented software
  - important given the wide adoption of OO technology
- Also relevant to mobile and agent-oriented systems
  - a device or process migrates and changes locality while retaining its connectivity (and its identity)
  - an agent has self-contained functionality and migrates to achieve efficiency gains while retaining references to its initiator and/or target

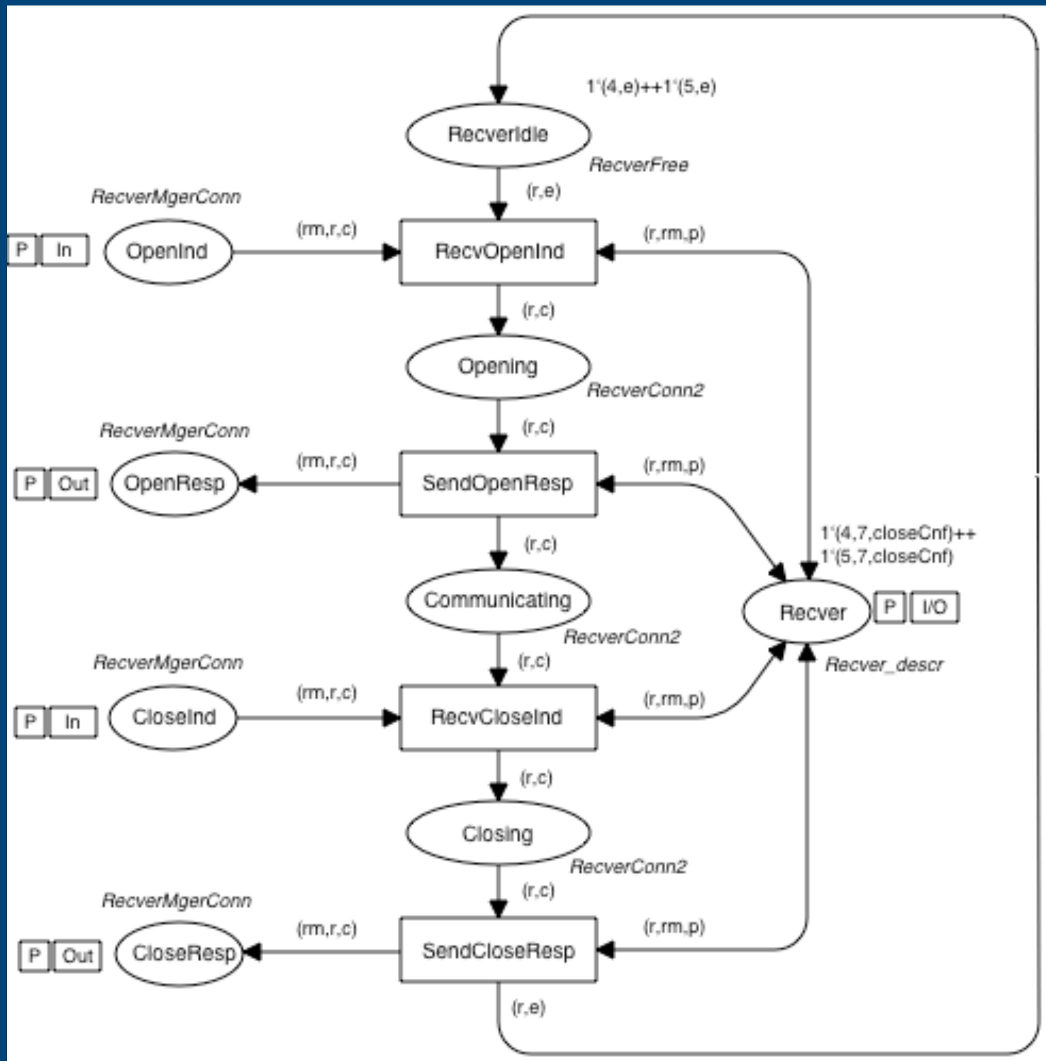
Example: Protocol for confirmed establishment of connections

- Sender



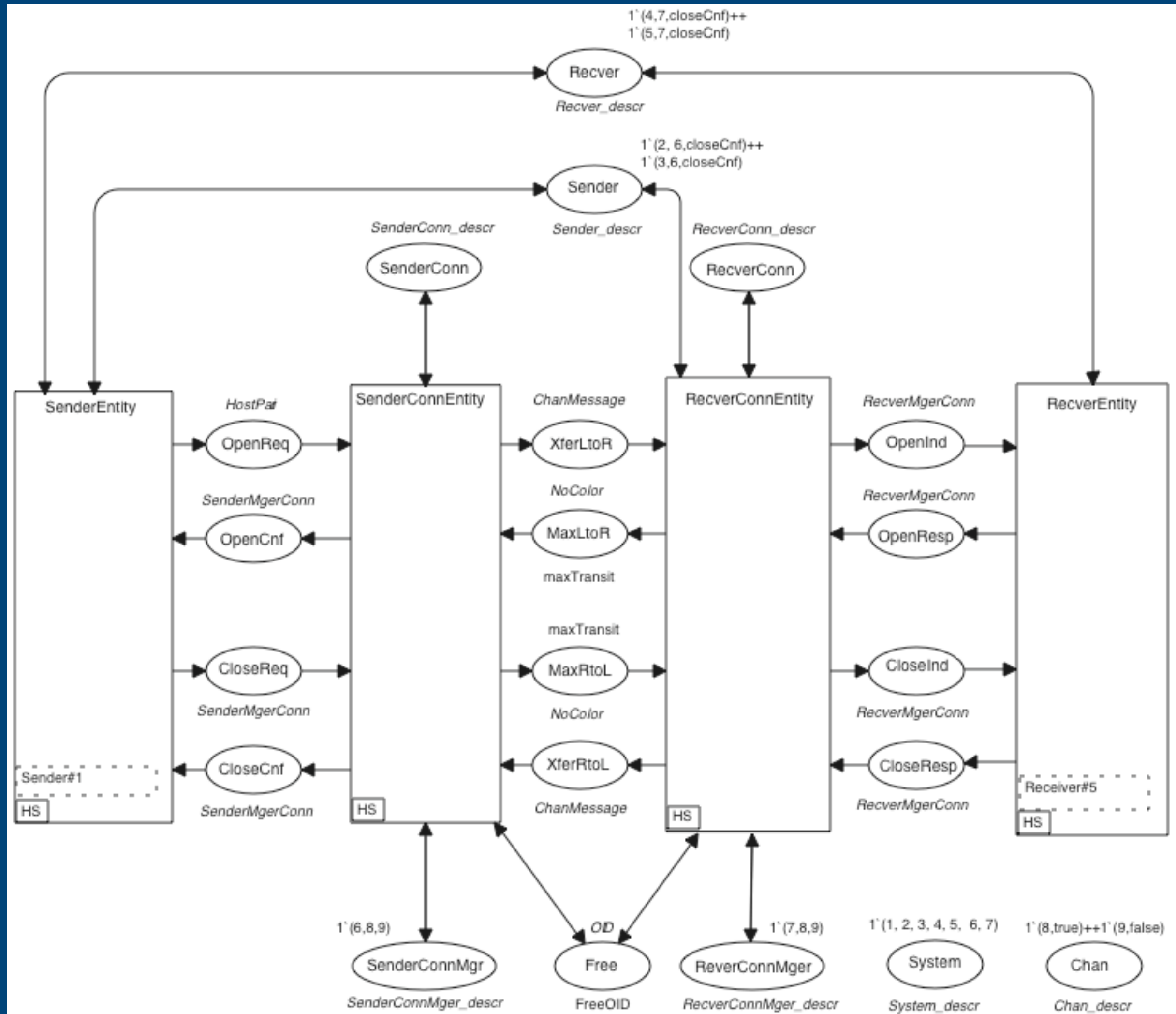
Example: Protocol for confirmed establishment of connections

- Receiver



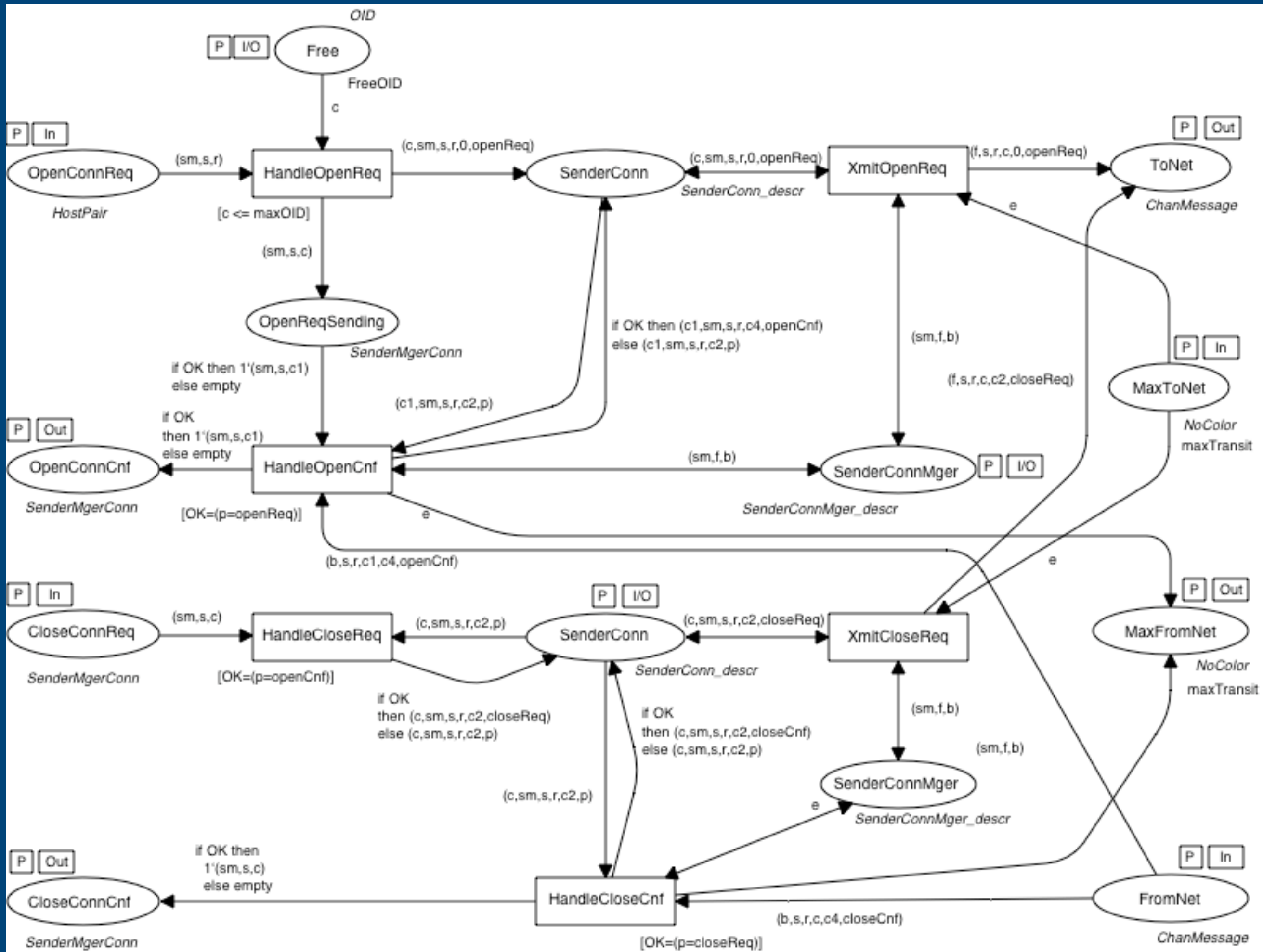
Example: Protocol for confirmed establishment of connections

- System



Example: Protocol for confirmed establishment of connections

- Sender manager



Notes on the protocol example

- All components are objects
  - senders, receivers, connection managers, connections
  - only connections are dynamically allocated and discarded
- The ***behaviour*** of an object is given by a Petri net
- The ***state*** of an object is given by its marking
- The ***identity*** of an object is given by an integer object identifier

- Multiple instances are folded onto the one subnet with object identifiers to distinguish the instances

State space exploration for object-based systems

- Basis of model-checking
- Primary obstacle is ***state space explosion***
- Need to adopt equivalence reduction so that states which are essentially the same are treated as such
- Need to eliminate garbage which could otherwise unnecessarily differentiate states

Raw state space results

Free = number of available identifiers
– each end of a connection requires one identifier

Identifiers allocated in any order

Identifiers allocated in sequence

	Full		
Free	Nodes	Arcs	Sec
3	9,897	28,716	9
4	256,617	826,540	1,794
3	3,153	8,064	2
4	48,725	155,680	89
5	87,029	284,272	403
6	>251,500		>1 hr

The dSPIN approach

- A depth-first traversal is performed of the system state
- Object identifiers are reallocated in the order of traversal
  - this produces a (unique) canonical representation
- The depth-first traversal is also used for garbage collection
  - a mark-and-sweep algorithm
- This approach does not deal with unordered collections of references
  - a general question of symmetry or graph isomorphism

Raw results and dSPIN algorithm (without enhancement)

	Full			$Canon_1$		
Free	Nodes	Arcs	Sec	Nodes	Arcs	Sec
3	9,897	28,716	9	1,538	4,532	1
4	256,617	826,540	1,794	7,524	24,896	7
5				8,982	32,806	10
6				10,712	42,820	13
7				10,720	48,268	14
16				10,720	96,940	24
3	3,153	8,064	2	1,538	4,480	1
4	48,725	155,680	89	7,524	24,096	7
5	87,029	284,272	403	8,982	30,786	9
6	$>$251,500		$>$1 hr	10,712	36,736	12
7				10,720	37,592	12
16				10,720	37,592	12

Sweepline method for reducing memory demands

- Applicable to systems which exhibit a notion of progress
  - states with an earlier progress value cannot be revisited from states with a later progress value
  - states with earlier progress values can be discarded
- The method can be extended to cater for ***regress*** edges
- E.g. protocol with numbered messages
- E.g. timed systems
- E.g. object-based systems

Experiments with three progress measures

- ψ_1 = next available (numeric) object identifier
- ψ_2 = weighted sum of connection progress (4 steps)
- ψ_3 = weighted sum of senders (16 steps)

	Full			Sweep-Line ψ_1		Sweep-Line ψ_2		Sweep-Line ψ_3	
Free	Nodes	Arcs	Sec	Peak	Sec	Peak	Sec	Peak	Sec
3	9,897	28,716	9	8,088	12	2,976	11	2,304	13
4	256,617	826,540	1,794	226,320	1,925	54,528	1,767	61,152	3,145
3	3,153	8,064	2	2,352	2	720	2	768	2
4	48,725	155,680	89	45,572	100	10,752	112	12,112	123
5	87,029	284,272	403	60,788	543	16,576	557	15,840	459
6	>251,500		>1 hr						

Combining equivalence reduction and sweepline

- Superficially contradictory
  - equivalence reduction looks to match already visited states
  - sweepline aims to avoid reconsidering prior states
- But examples commonly mix the two:
  - protocols with numbered messages may use cyclic numbering
  - timed systems may exhibit repeated patterns of behaviour
  - object-based systems may exhibit cyclic behaviour as objects are allocated and discarded

Combining equivalence reduction and sweepline

Free	Full			$Canon_1$			Combined $Canon_1, \psi_2$				Combined $Canon_1, \psi_3$			
	Nodes	Arcs	Sec	Nodes	Arcs	Sec	Nodes	Arcs	Peak	Sec	Nodes	Arcs	Peak	Sec
3	9,897	28,716	9	1,538	4,532	1	3,074	9,056	445	3	2,774	8,124	301	1
4	256,617	826,540	1,794	7,524	24,896	7	15,046	49,784	2,223	17	18,613	61,304	1,957	23
5				8,982	32,806	10	17,962	65,604	2,999	23	20,515	75,474	2,189	28
6				10,712	42,820	13	21,420	85,624	4,171	30	24,107	97,320	2,603	35
7				10,720	48,268	14	21,436	96,518	4,179	31	24,139	108,966	2,611	38
16				10,720	96,940	24	21,436	193,844	4,179	55	24,283	213,060	2,611	66
3	3,153	8,064	2	1,538	4,480	1	3,074	8,952	445	3	2,774	8,020	301	2
4	48,725	155,680	89	7,524	24,096	7	15,046	48,184	2,223	16	18,613	59,048	1,957	21
5	87,029	284,272	403	8,982	30,786	9	17,962	61,564	2,999	21	20,507	69,850	2,189	25
6	>251,500		>1 hr	10,712	36,736	12	21,420	73,458	4,171	25	24,091	82,222	2,603	30
7				10,720	37,592	12	21,436	75,170	4,179	27	24,107	83,562	2,611	31
16				10,720	37,592	12	21,436	75,170	4,169	27	24,107	83,562	2,611	31

Relating canonicalisation and progress

- Progress measures:
  - ψ_1 = next available (numeric) object identifier
  - ψ_2 = weighted sum of connection progress (4 steps)
  - ψ_3 = weighted sum of senders (16 steps)
- Canonicalisation functions:
  - Canon_1 = depth first traversal taking natural order of tokens in places
  - Canon_2 = order senders by progress measure ψ_3

Canonicalisation results

	Full			$Canon_1$			$Canon_2$		
Free	Nodes	Arcs	Sec	Nodes	Arcs	Sec	Nodes	Arcs	Sec
3	9,897	28,716	9	1,538	4,532	1	421	1,230	1
4	256,617	826,540	1,794	7,524	24,896	7	2,053	6,755	1
5				8,982	32,806	10	2,467	8,953	2
6				10,712	42,820	13	2,981	11,803	3
7				10,720	48,268	14	2,989	13,368	3
16				10,720	96,940	24	2,989	27,093	6
3	3,153	8,064	2	1,538	4,480	1	421	1,213	1
4	48,725	155,680	89	7,524	24,096	7	2,053	6,523	1
5	87,029	284,272	403	8,982	30,786	9	2,469	8,378	2
6	$>$251,500		$>$1 hr	10,712	36,736	12	2,981	10,116	3
7				10,720	37,592	12	2,989	10,408	3
16				10,720	37,592	12	2,989	10,408	3

Combined results for $Canon_2$

	Full			$Canon_2$			Combined $Canon_2$, ψ_2				Combined $Canon_2$, ψ_3			
Free	Nodes	Arcs	Sec	Nodes	Arcs	Sec	Nodes	Arcs	Peak	Sec	Nodes	Arcs	Peak	Sec
3	9,897	28,716	9	421	1,230	1	840	2,452	125	1	753	2,175	89	1
4	256,617	826,540	1,794	2,053	6,755	1	4,101	13,485	598	4	5,092	16,660	493	5
5				2,467	8,953	2	4,934	17,887	808	5	5,673	20,683	551	7
6				2,981	11,803	3	5,955	23,560	1,199	7	6,741	26,898	735	9
7				2,989	13,368	3	5,972	26,692	1,207	8	6,759	30,234	739	10
16				2,989	27,093	6	5,972	54,124	1,207	14	6,798	59,621	738	17
3	3,153	8,064	2	421	1,213	1	840	2,418	125	1	753	2,141	89	1
4	48,725	155,680	89	2,053	6,523	1	4,102	13,025	598	4	5,092	16,024	493	5
5	87,029	284,272	403	2,469	8,378	2	4,933	16,733	808	5	5,674	19,125	551	6
6	>251,500		>1 hr	2,981	10,116	3	5,956	20,198	1,199	6	6,741	22,763	735	8
7				2,989	10,408	3	5,971	20,778	1,207	7	6,756	23,250	739	8
16				2,989	10,408	3	5,971	20,778	1,207	7	6,756	23,250	739	8

Conclusions

- It is important to identify a good canonicalisation function for the state space exploration of object-based systems
  - in general, this is a difficult problem
- The sweepline method identifies a notion of progress
  - this can be used to conserve memory during state space exploration
  - it can also be used to define a canonicalisation function
- Results indicate that the same progress measure can be used for both purposes