University of South Australia

Position Paper:

On Extending the Sweep-Line for Language Equivalence Checking

Guy Edward Gallasch

Kindly presented by Lars Michael Kristensen

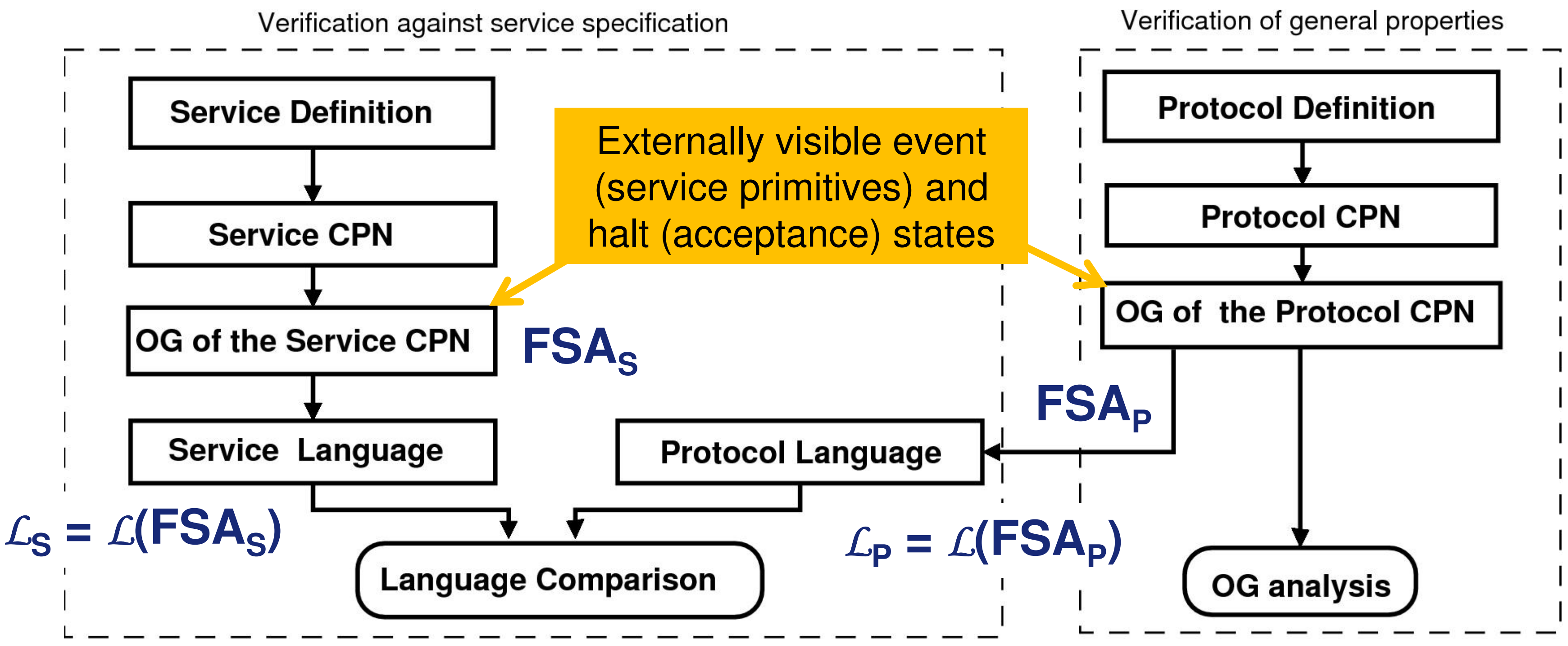
.. who made small and non-substantial modifications to this slide set that only he can be held responsible for…

Background and Motivation

University of South Australia

Protocol Verification Methodology:

1. **Verification of general properties: absence of deadlocks, livelocks…**
2. **Verification against its *service specification*.**



$$\mathcal{L}_S = \mathcal{L}_P \;?$$

University of South Australia

Background and Motivation

- **Question - language equivalence checking with sweep-line exploration?**
  - **Service language: $\mathcal{L}_{S.}$ Protocol language: $\mathcal{L}_P$ Equivalence: $\mathcal{L}_S = \mathcal{L}_P$**
  - **$\mathcal{L}_P \subseteq \mathcal{L}_S$: All behaviour of the protocol is allowed by the service specification.**
  - **$\mathcal{L}_S \subseteq \mathcal{L}_P$: The protocol implements (at least) all of its service.**

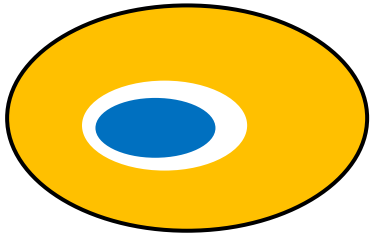
- **Previous work – protocol language inclusion checking, i.e., $\mathcal{L}_P \subseteq \mathcal{L}_{S:}$**
  - **All user-observable behaviour exhibited by the protocol is acceptable.**
  - **Acceptable in some circumstances, provided the protocol implements an acceptable subset of the service (determining this is a problem in its own right).**

- **This position paper - extension to language equivalence checking:**
  - **Can we simply check language inclusion and then the reverse, e.g. check $\mathcal{L}_P \subseteq \mathcal{L}_S$ and $\mathcal{L}_S \subseteq \mathcal{L}_P$ which would imply that $\mathcal{L}_S = \mathcal{L}_{P.}$**
  - **No! the OG of the protocol is prohibitively large, hence the use of sweep-line.**
  - **Requires extension of the sweep-line method with on-the-fly determinisation.**

University of South Australia

Language Inclusion Checking

- **Question - language equivalence checking with sweep-line exploration?**
  - **Service language: $\mathcal{L}_{S}$. Protocol language: $\mathcal{L}_P$ Equivalence: $\mathcal{L}_S = \mathcal{L}_P$**
  - **$\mathcal{L}_P \subseteq \mathcal{L}_S$: All behaviour of the protocol is allowed by the service specification.**
  - **$\mathcal{L}_S \subseteq \mathcal{L}_P$: The protocol implements (at least) all of its service.**
  - **Earlier work – protocol language inclusion checking, i.e., $\mathcal{L}_P \subseteq \mathcal{L}_S$:**
  - **All user-observable behaviour exhibited by the protocol is acceptable.**
  - **This holds if $\mathcal{L}(\overline{FSA_S}) \cap \mathcal{L}(FSA_P) = \mathcal{L}(\overline{FSA_S} \,||\, FSA_P)$ is empty:**

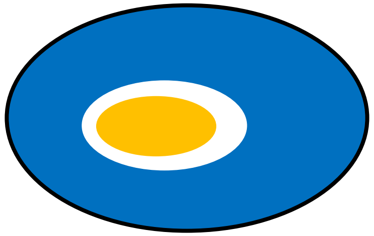


  - **FSA_S must be deterministic in order to obtain its complement.**
  - **Determinisation not usually a problem for FSA_S:**
    - **It is small enough to be known a priori, hence can be made deterministic and its complement found prior to exploring the protocol OG.**
    - **Both mapping from the OG to the FSA and the parallel composition of service and protocol FSAs can be performed on-the-fly with the sweep-line method.**

Language Equivalence Checking

- **This position paper - extension to language equivalence checking:**
  - **Can we simply check language inclusion and then the reverse, e.g. check $\mathcal{L}_P \subseteq \mathcal{L}_S$ and $\mathcal{L}_S \subseteq \mathcal{L}_P$ which would imply that $\mathcal{L}_S = \mathcal{L}_P$. ?**
  - **No! the OG of the protocol is prohibitively large (hence the use of sweep-line).**

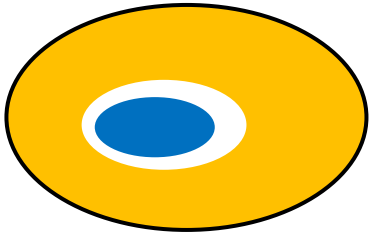
- **Checking $\mathcal{L}_S \subseteq \mathcal{L}_P$ - the protocol implements (at least) all of the service:**
  - **This holds if $\mathcal{L}(FSA_S) \cap \mathcal{L}(\overline{FSA_P}) = \mathcal{L}(FSA_S \,||\, \overline{FSA_P})$ is empty, :**



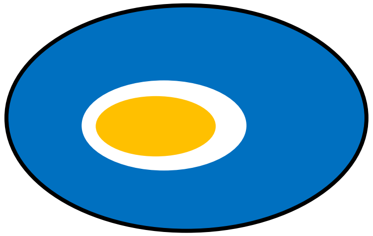
  - **This time FSA_P must be deterministic - but FSA_P (protocol OG) is large due to state explosion.**
  - **Determinisation, complement, and parallel composition must be done on-the-fly during sweep-line exploraion.**
  - **Requires extension of the sweep-line method with on-the-fly determinisation.**

University of South Australia

Language Equivalence Checking



- **Checking $\mathcal{L}_P \subseteq \mathcal{L}_S$ - protocol has not illegal behaviours:**
  - **This holds if $\mathcal{L}(\overline{FSA_S} \,||\, FSA_P)$ is empty.**
  - **FSA_S must be deterministic in order to obtain its complement.**
  - **Determinisation not usually a problem for FSA_S:**
    - **It is small enough to be known a priori, hence can be made deterministic and its complement found prior to exploring the protocol OG.**
    - **Both mapping from the OG to the FSA and the parallel composition of service and protocol FSAs can be performed on-the-fly with the sweep-line method.**
- **Checking $\mathcal{L}_S \subseteq \mathcal{L}_P$ - the protocol implements (at least) all of the service:**
  - **This holds if if $\mathcal{L}(FSA_S) \cap \mathcal{L}(\overline{FSA_P}) = \mathcal{L}(FSA_S \,||\, \overline{FSA_P})$ is empty,**



  - **but this time FSA_P must be deterministic.**
  - **The FSA_P (protocol OG) is large due to state explosion (which is why we want use the sweep-line method)**
  - **Determinisation, complement, and parallel composition must be done on-the-fly during sweep-line exploraion.**

Language Equivalence Checking

University of South Australia

- **Exactly what must we do on-the-fly?**
  1. **Map from the Protocol OG to FSA_P**
     1. **Arc labels map to service primitives or epsilon**
     2. **Recognise halt (acceptance) states**
  2. **Determinise FSA_P to produce $DFSA_P$**
  3. **Produce the complement of $DFSA_P$**
     1. **Introduce a "trap" state**
     2. **"complete" the FSA**
        (all states accept all symbols, leading to the trap state when not previously defined)
     3. **Invert halt states**

- **Mapping from the OG to the FSA (1) and producing the complement of $DFSA_P$ (3) can be done on a state-by-state and arc-by-arc basis.**

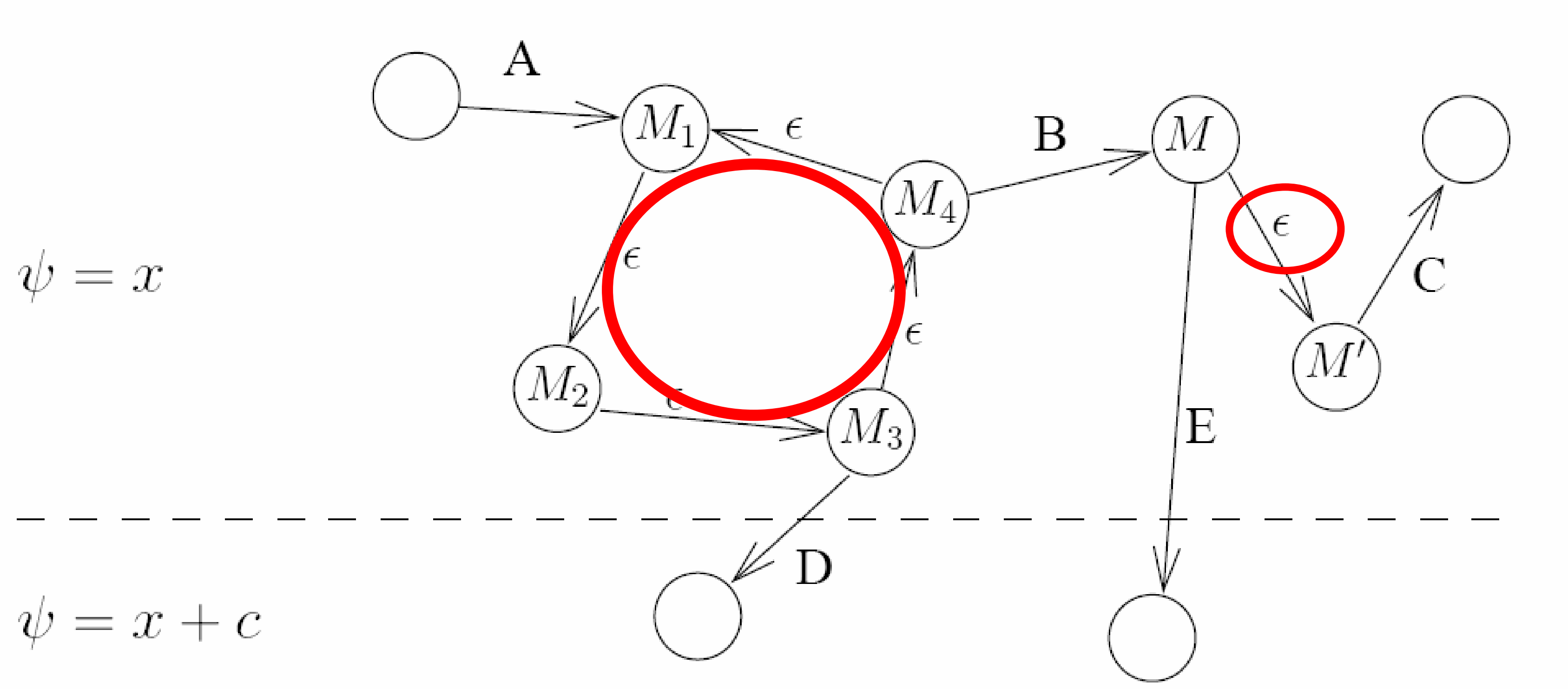
- **The non-trivial part is on-the-fly determinisation in presence of states being deleted from memory by the sweep-line method.**

University of South Australia

Language Equivalence Checking

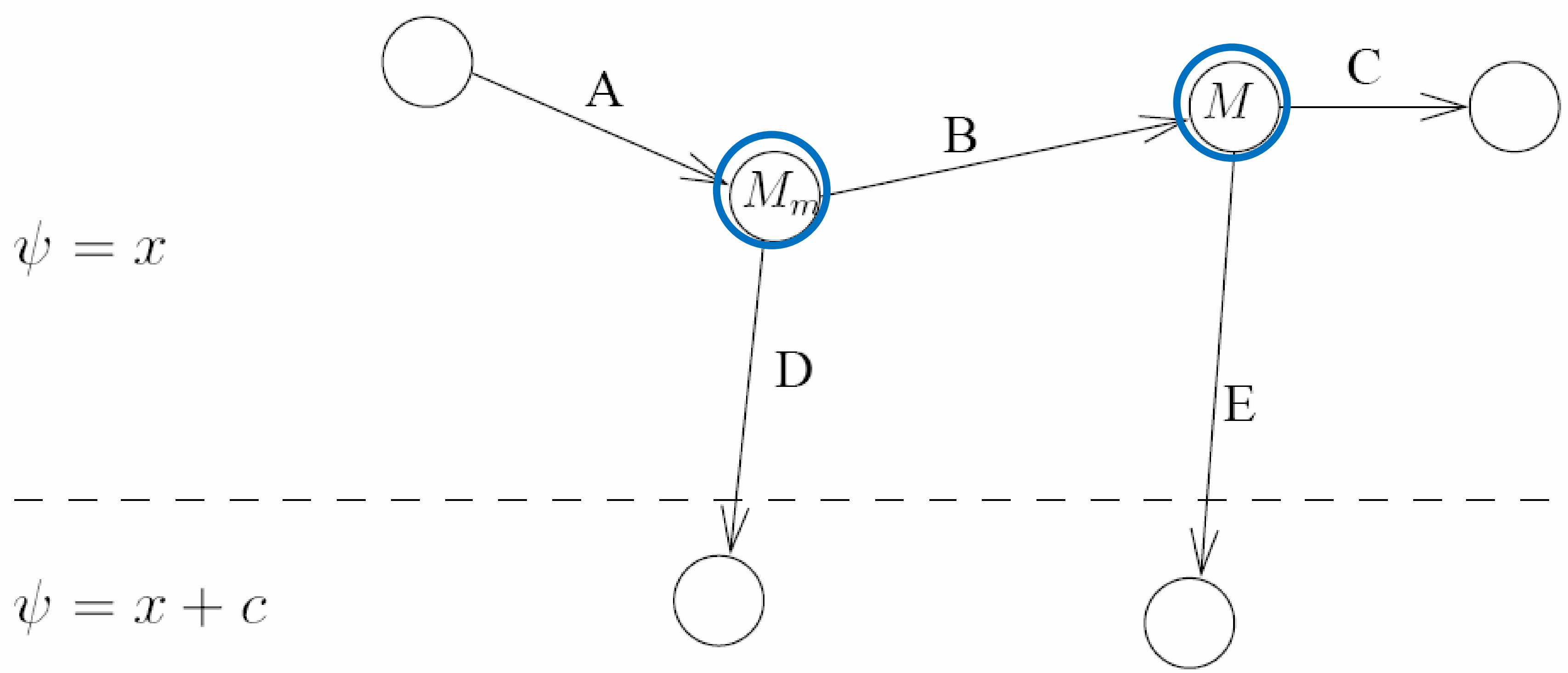
- **On-the-fly determinisation with the sweep-line method lends itself to a state-by-state (level-by-level) approach:**
  - **We adopt the techniques of e.g. Barrett and Couch [1]:**
    - **Remove empty (epsilon) cycles**
    - **Remove remaining empty (epsilon) moves**
    - **Remove remaining non-determinism**
  - **Challenge: empty cycles and empty moves may cross progress-level boundaries.**
  - **When it is safe for states to be deleted from memory?**
  - **Introduce transient states in addition to persistent states:**
    - **Persistent states: cannot be deleted (destinations of regress edges).**
    - **Transient states: must be retained in memory for now, for the purposes of determinisation, but can be deleted at some point in the future.**

Example: Some Simple Situations



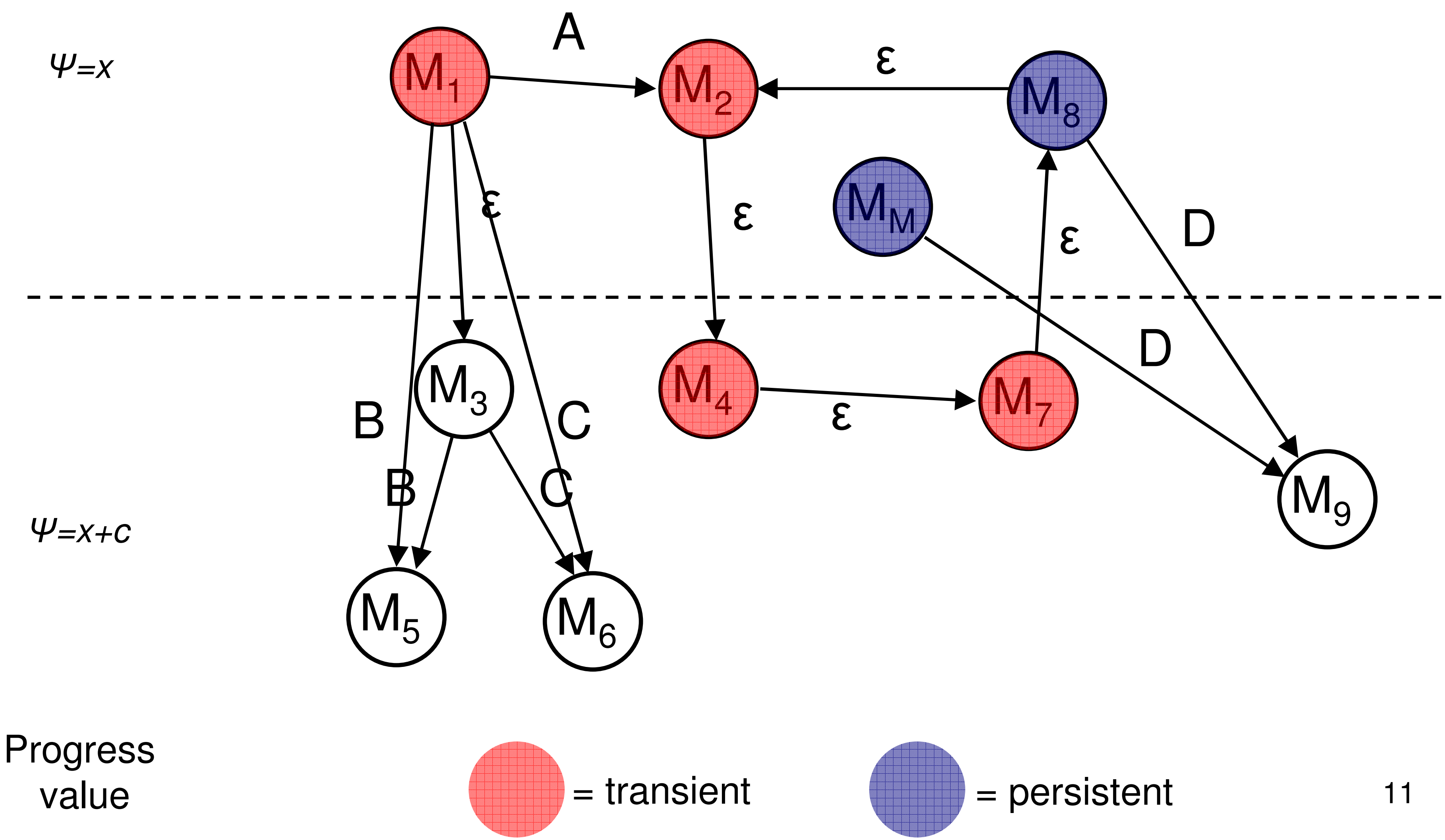
Progress value

Example: Some Simple Situations



Progress value

Example: Some Complex Situations



Conclusions and Future Work

University of South Australia

- This extension of the sweep-line method as presented in this position paper is at an early stage of development.
  - We have adopted the two-step approach as advocated by e.g. [1].
  - Much work remains to bring this work to fruition.
- The key is knowing when states must be temporarily retained for the purposes of determinisation, after which they can be deleted.
  - Not all examples are captured in the paper.
- It remains to:
  - Formalise the approach and verify its termination properties (currently a conjecture).
  - Develop a modified sweep-line exploration algorithm that takes into account on-the-fly determinisation.
  - Evaluate its effectiveness on a substantial case study.
- Determinisation can be interleaved with the complementation and parallel composition activities
  - It remains to be seen whether full interleaving of these activities or "batch" processing is more efficient/effective.

Conclusions and Future Work

University of South Australia

- We have adopted the two-step approach as advocated by e.g. [1].
- Determinisation via the power set/subset construction technique (e.g. [15]) with lazy subset evaluation is another approach that may be investigated
  - States of the deterministic FSA are power sets of the set of global states.
  - “Lazy” evaluation: calculate subsets (epsilon-closures) only as they are required.
  - Intuitively, exploring all successors reachable via epsilon moves is less suited to Sweep-line analysis, as this may frequently violate the Sweep-line’s “least-progress-first” exploration policy.
  - It should be possible to defer the calculation of epsilon-closures, however this results in an algorithm that looks remarkably similar to the one already adopted.

- If our goal is simply to detect a violation of language equivalence, it may be possible to take a more direct approach in some cases:
  - The parallel composition of FSA_S and FSA_P (not its complement) can be built on-the-fly and the symbols accepted by each state compared with the corresponding states in FSA_S and FSA_P.
  - If they don’t match, we have a violation of language equivalence.
  - On-the-fly complementation is no longer required, however, the hard work of determinisation still is.

University of South Australia

A Final Acknowledgement

I would like to sincerely thank Lars Kristensen for the discussions we held on this topic many years ago, and especially for agreeing to present this on my behalf.

Thankyou Lars!

Questions?