A Synchronous Effects Logic for Temporal Verification of Pure Esterel

Yahui Song and Wei-Ngan Chin

School of Computing, NUS

@VMCAI2021, 19 January 2021

A Synchronous Effects Logic for Temporal Verification of Pure Esterel

Yahui Song and Wei-Ngan Chin

School of Computing, NUS

@VMCAI2021, 19 January 2021

VMCAI Artifact Evaluation Functional

Esterel – A synchronous language

- System-design language/modelling language.

```
signal S1 in
  present S1
    then emit S1
    else nothing
  end present
end signal
```

Esterel – A synchronous language

- System-design language/modelling language.
- Deterministic semantics.

```
signal S1 in
  present S1
    then emit S1
    else nothing
  end present
end signal
```

Esterel – A synchronous language

- System-design language/modelling language.
- Deterministic semantics.
- Primitive constructs execute in zero time except for the pause statement.

```
signal S1 in
  present S1
    then emit S1
    else nothing
  end present
end signal
```

Esterel – A synchronous language

- System-design language/modelling language.
- Deterministic semantics.
- Primitive constructs execute in zero time except for the pause statement.
- The (i) correctness and (ii) safety issues are particularly critical.

```
signal S1 in
  present S1
    then emit S1
    else nothing
  end present
end signal
```

Esterel – A synchronous language

- System-design language/modelling language.
- Deterministic semantics.
- Primitive constructs execute in zero time except for the pause statement.
- The (i) correctness and (ii) safety issues are particularly critical.

```
signal S1 in
  present S1
    then emit S1
    else nothing
  end present
end signal
```

Logically incorrect

two possible assignments to S1.

S1 can be both present or absent.

Overview (1)

Synced Effects – the specification language

- Specify the temporal properties

  into the pre/post condition.

```
module close:
output CLOSE;
/*@ requires {OPEN}
    ensures {}.{CLOSE} @*/
  pause; emit CLOSE
end module
```

Fig. 1. The close module

Overview (1)

Synced Effects – the specification language

- Specify the temporal properties

  into the pre/post condition.

```
module close:
output CLOSE;
/*@ requires {OPEN}
    ensures {}.{CLOSE} @*/
  pause; emit CLOSE
end module
```

Fig. 1. The close module

Overview (1)

Synced Effects – the specification language

- Specify the temporal properties into the pre/post condition.

```
module close:
output CLOSE;
/*@ requires {OPEN}
    ensures {}.{CLOSE} @*/
  pause; emit CLOSE
end module
```

Fig. 1. The close module

```
module manager:
input BTN;
output CLOSE;
/*@
requires {}
ensures ({BTN}.{CLOSE}\/{})*
@*/
  signal OPEN in
    loop
      emit OPEN;
      present BTN
        then run close
        else nothing
      end present;
      pause
    end loop
  end signal
end module
```

Fig. 2. The manager module

Overview (2)

The Forward Verifier –
To accumulate the effects

1) loop
$\langle\{\}\rangle$

2) emit OPEN;
$\langle\{\mathtt{OPEN}\}\rangle$ **[FV-Emit]**

3) present BTN then
$\langle\{\mathtt{OPEN}, \mathtt{BTN}\}\rangle$ **[FV-Present]**

4) run close
$\{\mathtt{OPEN}, \mathtt{BTN}\} \sqsubseteq \{\mathtt{OPEN}\}$ *(-TRS: check precondition, succeed-)*
$\langle\{\mathtt{OPEN}, \mathtt{BTN}\} \cdot \{\mathtt{CLOSE}\}\rangle$ **[FV-Call]**

5) else nothing
$\langle\{\mathtt{OPEN}\}\rangle$ **[FV-Present]**

6) end present;
$\langle\{\mathtt{OPEN}, \mathtt{BTN}\} \cdot \{\mathtt{CLOSE}\} \vee \{\mathtt{OPEN}\}\rangle$ **[FV-Present]**

7) pause
$\langle(\{\mathtt{OPEN}, \mathtt{BTN}\} \cdot \{\mathtt{CLOSE}\} \vee \{\mathtt{OPEN}\}) \cdot \{\}\rangle$ **[FV-Pause]**

8) end loop
$\langle(\{\mathtt{OPEN}, \mathtt{BTN}\} \cdot \{\mathtt{CLOSE}\} \vee \{\mathtt{OPEN}\})^\star\rangle$ **[FV-Loop]**

9) $(\{\mathtt{OPEN}, \mathtt{BTN}\} \cdot \{\mathtt{CLOSE}\} \vee \{\mathtt{OPEN}\})^\star \sqsubseteq (\{\mathtt{BTN}\} \cdot \{\mathtt{CLOSE}\} \vee \{\})^\star$ *(-TRS: check postcondition, succeed-)*

Overview (2)

The Forward Verifier –
To accumulate the effects

Add the events into the effect state

1) loop
$\langle\{\}\rangle$

2) emit OPEN;
$\langle\{\texttt{OPEN}\}\rangle$ **[FV-Emit]**

3) present BTN then
$\langle\{\texttt{OPEN}, \texttt{BTN}\}\rangle$ **[FV-Present]**

4) run close
$\{\texttt{OPEN}, \texttt{BTN}\} \sqsubseteq \{\texttt{OPEN}\}$ *(-TRS: check precondition, succeed-)*
$\langle\{\texttt{OPEN}, \texttt{BTN}\} \cdot \{\texttt{CLOSE}\}\rangle$ **[FV-Call]**

5) else nothing
$\langle\{\texttt{OPEN}\}\rangle$ **[FV-Present]**

6) end present;
$\langle\{\texttt{OPEN}, \texttt{BTN}\} \cdot \{\texttt{CLOSE}\} \vee \{\texttt{OPEN}\}\rangle$ **[FV-Present]**

7) pause
$\langle(\{\texttt{OPEN}, \texttt{BTN}\} \cdot \{\texttt{CLOSE}\} \vee \{\texttt{OPEN}\}) \cdot \{\}\rangle$ **[FV-Pause]**

8) end loop
$\langle(\{\texttt{OPEN}, \texttt{BTN}\} \cdot \{\texttt{CLOSE}\} \vee \{\texttt{OPEN}\})^\star\rangle$ **[FV-Loop]**

9) $(\{\texttt{OPEN}, \texttt{BTN}\} \cdot \{\texttt{CLOSE}\} \vee \{\texttt{OPEN}\})^\star \sqsubseteq (\{\texttt{BTN}\} \cdot \{\texttt{CLOSE}\} \vee \{\})^\star$ *(-TRS: check postcondition, succeed-)*

Overview (2)

The Forward Verifier – To accumulate the effects

1) loop
$\langle\{\}\rangle$

2) emit OPEN;
$\langle\{\texttt{OPEN}\}\rangle$ [FV-Emit]

3) present BTN then
$\langle\{\texttt{OPEN},\texttt{BTN}\}\rangle$ [FV-Present]

4) run close
$\{\texttt{OPEN},\texttt{BTN}\} \sqsubseteq \{\texttt{OPEN}\}$ *(-TRS: check precondition, succeed-)*
$\langle\{\texttt{OPEN},\texttt{BTN}\} \cdot \{\texttt{CLOSE}\}\rangle$ [FV-Call]

5) else nothing
$\langle\{\texttt{OPEN}\}\rangle$ [FV-Present]

6) end present;
$\langle\{\texttt{OPEN},\texttt{BTN}\} \cdot \{\texttt{CLOSE}\} \vee \{\texttt{OPEN}\}\rangle$ [FV-Present]

7) pause
$\langle(\{\texttt{OPEN},\texttt{BTN}\} \cdot \{\texttt{CLOSE}\} \vee \{\texttt{OPEN}\}) \cdot \{\}\rangle$ [FV-Pause]

8) end loop
$\langle(\{\texttt{OPEN},\texttt{BTN}\} \cdot \{\texttt{CLOSE}\} \vee \{\texttt{OPEN}\})^\star\rangle$ [FV-Loop]

9) $(\{\texttt{OPEN},\texttt{BTN}\} \cdot \{\texttt{CLOSE}\} \vee \{\texttt{OPEN}\})^\star \sqsubseteq (\{\texttt{BTN}\} \cdot \{\texttt{CLOSE}\} \vee \{\})^\star$ *(-TRS: check postcondition, succeed-)*

Add the events into the effect state

Check if the current effect satisfies the callee's precondition

Overview (2)

The Forward Verifier – To accumulate the effects

1) loop
$\langle\{\}\rangle$

2) emit OPEN;
$\langle\{\texttt{OPEN}\}\rangle$ [FV-Emit]

Add the events into the effect state

3) present BTN then
$\langle\{\texttt{OPEN}, \texttt{BTN}\}\rangle$ [FV-Present]

4) run close
$\{\texttt{OPEN}, \texttt{BTN}\} \sqsubseteq \{\texttt{OPEN}\}$ *(-TRS: check precondition, succeed-)*
$\langle\{\texttt{OPEN}, \texttt{BTN}\} \cdot \{\texttt{CLOSE}\}\rangle$ [FV-Call]

Check if the current effect satisfies the callee's precondition

5) else nothing
$\langle\{\texttt{OPEN}\}\rangle$ [FV-Present]

6) end present;
$\langle\{\texttt{OPEN}, \texttt{BTN}\} \cdot \{\texttt{CLOSE}\} \vee \{\texttt{OPEN}\}\rangle$ [FV-Present]

7) pause
$\langle(\{\texttt{OPEN}, \texttt{BTN}\} \cdot \{\texttt{CLOSE}\} \vee \{\texttt{OPEN}\}) \cdot \{\}\rangle$ [FV-Pause]

8) end loop
$\langle(\{\texttt{OPEN}, \texttt{BTN}\} \cdot \{\texttt{CLOSE}\} \vee \{\texttt{OPEN}\})^{\star}\rangle$ [FV-Loop]

9) $(\{\texttt{OPEN}, \texttt{BTN}\} \cdot \{\texttt{CLOSE}\} \vee \{\texttt{OPEN}\})^{\star} \sqsubseteq (\{\texttt{BTN}\} \cdot \{\texttt{CLOSE}\} \vee \{\})^{\star}$ *(-TRS: check postcondition, succeed-)*

Checks if the final effects satisfy the Program's postcondition

Overview (3)

Term Rewriting System –
the Effects inclusion checker

$$
\dfrac{
\dfrac{
\dfrac{
\dfrac{\Phi \sqsubseteq \Phi_{\texttt{post}} (\dagger) \quad [\text{REOCCUR}]}{\mathcal{E} \cdot \Phi \sqsubseteq (\mathcal{E} \vee \bot) \cdot \Phi_{\texttt{post}}}
}{\{\texttt{CLOSE}\} \cdot \Phi \sqsubseteq (\{\texttt{CLOSE}\} \vee \mathcal{E}) \cdot \Phi_{\texttt{post}}} \; [\text{UNFOLD}]
}{\{\texttt{OPEN}, \texttt{BTN}\} \cdot \{\texttt{CLOSE}\} \cdot \Phi \sqsubseteq \Phi_{\texttt{post}}} \; [\text{UNFOLD}]
\qquad
\dfrac{
\dfrac{\Phi \sqsubseteq \Phi_{\texttt{post}} (\dagger) \quad [\text{REOCCUR}]}{\mathcal{E} \cdot \Phi \sqsubseteq (\bot \vee \mathcal{E}) \cdot \Phi_{\texttt{post}}} \; [\text{UNFOLD}]
}{\{\texttt{OPEN}\} \cdot \Phi \sqsubseteq \Phi_{\texttt{post}}} \; [\text{UNFOLD}]
}{\Phi \sqsubseteq \Phi_{\texttt{post}} (\dagger)}
$$

$$\text{where} \quad \Phi = (\{\texttt{OPEN}, \texttt{BTN}\} \cdot \{\texttt{CLOSE}\} \vee \{\texttt{OPEN}\})^{\star}; \quad \text{and} \quad \Phi_{\texttt{post}} = (\{\texttt{BTN}\} \cdot \{\texttt{CLOSE}\} \vee \{\})^{\star}$$

Overview (3)

Term Rewriting System –
the Effects inclusion checker

$$
\dfrac{
\dfrac{
\dfrac{
\dfrac{\Phi \sqsubseteq \Phi_{\mathtt{post}}\,(\dagger) \quad [\text{REOCCUR}]}
{\mathcal{E} \cdot \Phi \sqsubseteq (\mathcal{E} \vee \bot) \cdot \Phi_{\mathtt{post}}}
}{\{\mathtt{CLOSE}\} \cdot \Phi \sqsubseteq (\{\mathtt{CLOSE}\} \vee \mathcal{E}) \cdot \Phi_{\mathtt{post}}}\ [\text{UNFOLD}]
}{\{\mathtt{OPEN}, \mathtt{BTN}\} \cdot \{\mathtt{CLOSE}\} \cdot \Phi \sqsubseteq \Phi_{\mathtt{post}}}\ [\text{UNFOLD}]
\qquad
\dfrac{
\dfrac{\Phi \sqsubseteq \Phi_{\mathtt{post}}\,(\dagger) \quad [\text{REOCCUR}]}
{\mathcal{E} \cdot \Phi \sqsubseteq (\bot \vee \mathcal{E}) \cdot \Phi_{\mathtt{post}}}\ [\text{UNFOLD}]
}{\{\mathtt{OPEN}\} \cdot \Phi \sqsubseteq \Phi_{\mathtt{post}}}\ [\text{UNFOLD}]
}{\Phi \sqsubseteq \Phi_{\mathtt{post}}\,(\dagger)}
$$

$$\text{where} \quad \Phi = (\{\mathtt{OPEN}, \mathtt{BTN}\} \cdot \{\mathtt{CLOSE}\} \vee \{\mathtt{OPEN}\})^{\star}; \quad \text{and} \quad \Phi_{\mathtt{post}} = (\{\mathtt{BTN}\} \cdot \{\mathtt{CLOSE}\} \vee \{\})^{\star}$$

Overview (3)

Term Rewriting System – the Effects inclusion checker

$$
\dfrac{
\dfrac{
\dfrac{
\dfrac{\Phi \sqsubseteq \Phi_{\mathtt{post}}\,(\dagger)\quad[\text{REOCCUR}]}{\mathcal{E}\cdot\Phi \sqsubseteq (\mathcal{E}\vee\bot)\cdot\Phi_{\mathtt{post}}}
}{\{\mathtt{CLOSE}\}\cdot\Phi \sqsubseteq (\{\mathtt{CLOSE}\}\vee\mathcal{E})\cdot\Phi_{\mathtt{post}}}\,[\text{UNFOLD}]
}{\{\mathtt{OPEN},\mathtt{BTN}\}\cdot\{\mathtt{CLOSE}\}\cdot\Phi \sqsubseteq \Phi_{\mathtt{post}}}\,[\text{UNFOLD}]
\qquad
\dfrac{
\dfrac{\Phi \sqsubseteq \Phi_{\mathtt{post}}\,(\dagger)\quad[\text{REOCCUR}]}{\mathcal{E}\cdot\Phi \sqsubseteq (\bot\vee\mathcal{E})\cdot\Phi_{\mathtt{post}}}\,[\text{UNFOLD}]
}{\{\mathtt{OPEN}\}\cdot\Phi \sqsubseteq \Phi_{\mathtt{post}}}\,[\text{UNFOLD}]
}{\Phi \sqsubseteq \Phi_{\mathtt{post}}\,(\dagger)}
$$

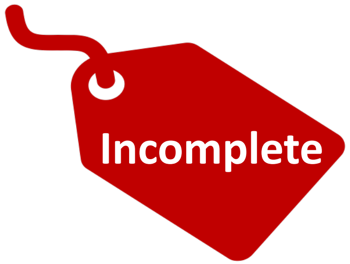
$$
\text{where}\quad \Phi = (\{\mathtt{OPEN},\mathtt{BTN}\}\cdot\{\mathtt{CLOSE}\}\vee\{\mathtt{OPEN}\})^{\star};\quad \text{and}\quad \Phi_{\mathtt{post}}=(\{\mathtt{BTN}\}\cdot\{\mathtt{CLOSE}\}\vee\{\})^{\star}
$$

Why Synchronous Effects?

Any Benefits?

Correctness checking &Temporal verification

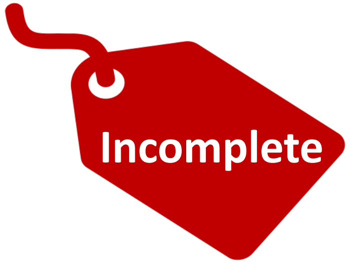
- Logical Correctness and Constructiveness checking:



  - Different semantics of Esterel;
  - Can not deal with unbounded input signals

Correctness checking &Temporal verification

- Logical Correctness and Constructiveness checking:
  - Different semantics of Esterel;
  - Can not deal with unbounded input signals



1) present S1 $\langle\{\}\rangle$
2) then nothing $\langle\{S1 \wedge \overline{S1}\}\rangle$
3) else emit S1 $\langle\{\overline{S1} \wedge S1\}\rangle$
4) end present $\langle\{false\} \vee \{false\}\rangle$

false $\rightarrow$ logical incorrect

Fig. 12.

Correctness checking &Temporal verification

- Logical Correctness and Constructiveness checking: **Incomplete**
  - Different semantics of Esterel;
  - Can not deal with unbounded input signals
- Temporal verification: **Low Efficiency**
  - Given an LTL formula;
  - Recursively translate it into an Esterel program that violate the safety formula;
  - Compose it in parallel with the given Esterel program to be verified;

Correctness checking &Temporal verification

While in our method:

1) No need to translate temporal properties into automata.

2) Disprove entailments earlier.

   - The [Nullable] rule

3) Scalable expressiveness for temporal properties.

   - $n > 0 \wedge \{A\}.\{A\}^{n-1} \mid\!- \{A\}^{n}$

- Logical Correctness and
  - Different semantics of E
  - Can not deal with unbo
- Temporal verification:

  Low Efficiency

  - Given an LTL formula;
  - Recursively translate it into an Esterel program that violate the safety formula;
  - Compose it in parallel with the given Esterel program to be verified;

Correctness checking &Temporal verification

- Logical Correctness and Constructiveness checking: **Incomplete**
  - Different semantics of Esterel;
  - Can not deal with unbounded input signals
- Temporal verification: **Low Efficiency**
  - Given an LTL formula;
  - Recursively translate it into an Esterel program that violate the safety formula;
  - Compose it in parallel with the given Esterel program to be verified;

Implementation and Evaluation

- An open-sourced prototype system using Ocaml.
- Benchmarks:
  1. CEC: It is an open-source compiler which provides pure Esterel programs for testing.
  2. Hiphop.js: It is a DSL for JavaScript. We take a subset of Hiphop.js programs and translate them into our target language.
- 96 pure Esterel programs, (10 ~ 300 lines). We manually annotate temporal in synced effects for each of them, including both succeeded and failed instances.

Summary

- **The Synced Effects :** We define the syntax and semantics of the Synced Effects.
- **Automated Verification System :** Targeting a pure Esterel language we develop:
  1) a Hoare-style forward verifier; and
  2) an effects inclusion checker (the TRS).
- **A prototype system of the novel effects logic:** Proven to be sound, with experimental results and case studies to show the feasibility.

Summary

Thanks a lot for your attention!

- **The Synced Effects :** We define the syntax and semantics of the Synced Effects.
- **Automated Verification System :** Targeting a pure Esterel language we develop:
    1) a Hoare-style forward verifier; and
    2) an effects inclusion checker (the TRS).
- **A prototype system of the novel effects logic:** Proven to be sound, with experimental results and case studies to show the feasibility.