Unlocking Blocked Communicating Processes

Adrian Francalanza[1] Marco Giunti[2] Antonio Ravara[3]

1. University of Malta
2. Universidade da Beira Interior
3. Universidade NOVA de Lisboa

12 April 2015

(Dead)locks and (Dead)lock-Free

$$\mathrm{in}(a, P) \stackrel{\text{def}}{=} P \equiv P' \parallel a.P'' \qquad \mathrm{out}(a, P) \stackrel{\text{def}}{=} P \equiv P' \parallel \bar{a}.P''$$

$$\mathrm{wait}(a, P) \stackrel{\text{def}}{=} \mathrm{in}(a, P) \ \mathrm{exor} \ \mathrm{out}(a, P)$$

$$\mathrm{sync}(a, P) \stackrel{\text{def}}{=} \mathrm{in}(a, P) \ \mathrm{and} \ \mathrm{out}(a, P)$$

(Dead)locks and (Dead)lock-Free

$$\mathrm{in}(a,P) \stackrel{\text{def}}{=} P \equiv P' \parallel a.P'' \qquad \mathrm{out}(a,P) \stackrel{\text{def}}{=} P \equiv P' \parallel \bar{a}.P''$$

$$\mathrm{wait}(a,P) \stackrel{\text{def}}{=} \mathrm{in}(a,P) \ \text{exor} \ \mathrm{out}(a,P)$$

$$\mathrm{sync}(a,P) \stackrel{\text{def}}{=} \mathrm{in}(a,P) \ \text{and} \ \mathrm{out}(a,P)$$

$$\mathrm{dlock}(P) \stackrel{\text{def}}{=} P \not\longrightarrow \quad \text{and} \quad \exists\, a.\mathrm{wait}(a,P)$$

$$\mathrm{dlfree}(P) \stackrel{\text{def}}{=} \left(P \longrightarrow^{*} Q \text{ and } Q \not\longrightarrow\right) \ \text{implies} \ \ \forall\, a.\neg\mathrm{wait}(a,Q)$$

(Dead)locks and (Dead)lock-Free

$$\mathrm{in}(a,P) \stackrel{\text{def}}{=} P \equiv P' \parallel a.P'' \qquad \mathrm{out}(a,P) \stackrel{\text{def}}{=} P \equiv P' \parallel \bar{a}.P''$$

$$\mathrm{wait}(a,P) \stackrel{\text{def}}{=} \mathrm{in}(a,P) \ \text{exor} \ \mathrm{out}(a,P)$$

$$\mathrm{sync}(a,P) \stackrel{\text{def}}{=} \mathrm{in}(a,P) \ \text{and} \ \mathrm{out}(a,P)$$

$$\mathrm{dlock}(P) \stackrel{\text{def}}{=} P \not\longrightarrow \quad \text{and} \quad \exists\, a.\mathrm{wait}(a,P)$$

$$\mathrm{dlfree}(P) \stackrel{\text{def}}{=} \left(P \longrightarrow^* Q \text{ and } Q \not\longrightarrow\right) \quad \text{implies} \quad \forall\, a.\neg\mathrm{wait}(a,Q)$$

$$\mathrm{lock}(P) \stackrel{\text{def}}{=} \mathrm{wait}(a,P) \quad \text{and} \quad \forall Q.\ P \longrightarrow^* Q \text{ implies } \neg\mathrm{sync}(a,Q)$$

$$\mathrm{lfree}(P) \stackrel{\text{def}}{=} \left\{ \begin{array}{l} \left(P \longrightarrow^* Q \text{ and } \mathrm{wait}(a,Q)\right) \text{ implies} \\ \qquad\qquad \exists R.\ Q \longrightarrow^* R \text{ and } \mathrm{sync}(a,R) \end{array} \right.$$

Deadlocks and Locks

Example

1. **0** and Ω (dead) lock-free

Deadlocks and Locks

Example

1. $\mathbf{0}$ and Ω (dead) lock-free
2. $a.\mathbf{0}$ and $\bar{a}.\mathbf{0}$ deadlocked... (thus locked)

Deadlocks and Locks

Example

1. $\mathbf{0}$ and Ω (dead) lock-free
2. $a.\mathbf{0}$ and $\bar{a}.\mathbf{0}$ deadlocked... (thus locked)
3. $\Omega \| a.\mathbf{0}$ and $\Omega \| \bar{a}.\mathbf{0}$ locked BUT NOT deadlocked

Deadlocks and Locks

Example

1. $\mathbf{0}$ and Ω (dead) lock-free
2. $a.\mathbf{0}$ and $\bar{a}.\mathbf{0}$ deadlocked... (thus locked)
3. $\Omega \| a.\mathbf{0}$ and $\Omega \| \bar{a}.\mathbf{0}$ locked BUT NOT deadlocked
4. $a.\bar{a}.\mathbf{0}$ deadlocked

Deadlocks and Locks

Example

1. $\mathbf{0}$ and Ω (dead) lock-free
2. $a.\mathbf{0}$ and $\bar{a}.\mathbf{0}$ deadlocked... (thus locked)
3. $\Omega \| a.\mathbf{0}$ and $\Omega \| \bar{a}.\mathbf{0}$ locked BUT NOT deadlocked
4. $a.\bar{a}.\mathbf{0}$ deadlocked
5. $b.\mathbf{0} \| \bar{b}.a.\bar{a}.\mathbf{0}$ potentially (and eventually) deadlocked

Deadlocks and Locks

Example

1. $\mathbf{0}$ and Ω (dead) lock-free
2. $a.\mathbf{0}$ and $\bar{a}.\mathbf{0}$ deadlocked... (thus locked)
3. $\Omega \| a.\mathbf{0}$ and $\Omega \| \bar{a}.\mathbf{0}$ locked BUT NOT deadlocked
4. $a.\bar{a}.\mathbf{0}$ deadlocked
5. $b.\mathbf{0} \| \bar{b}.a.\bar{a}.\mathbf{0}$ potentially (and eventually) deadlocked
6. $a.\bar{b}.\mathbf{0} \| b.\bar{a}.\mathbf{0}$ deadlocked

Deadlocks and Locks

Example

1. $\mathbf{0}$ and Ω (dead) lock-free
2. $a.\mathbf{0}$ and $\bar{a}.\mathbf{0}$ deadlocked... (thus locked)
3. $\Omega \| a.\mathbf{0}$ and $\Omega \| \bar{a}.\mathbf{0}$ locked BUT NOT deadlocked
4. $a.\bar{a}.\mathbf{0}$ deadlocked
5. $b.\mathbf{0} \parallel \bar{b}.a.\bar{a}.\mathbf{0}$ potentially (and eventually) deadlocked
6. $a.\bar{b}.\mathbf{0} \parallel b.\bar{a}.\mathbf{0}$ deadlocked
7. $a.(\bar{b}.\mathbf{0} \| c.\mathbf{0}) \parallel b.\bar{a}.\mathbf{0} \parallel \bar{c}.\mathbf{0}$ deadlocked

Type Systems for (dead)Lock Freedom

- Most works focus on guaranteeing (dead)lock freedom.
- Some work that deals with resolving (dead)locks.

Type Systems for (dead)Lock Freedom

- Most works focus on guaranteeing (dead)lock freedom.
- Some work that deals with resolving (dead)locks.

Example (Catalysers)

$$a.\bar{a}.\mathbf{0}$$

Type Systems for (dead)Lock Freedom

- Most works focus on guaranteeing (dead)lock freedom.
- Some work that deals with resolving (dead)locks.

Example (Catalysers)

$$a.\bar{a}.\mathbf{0} \parallel \underbrace{\bar{a}.a.\mathbf{0}}_{\text{catalyser code}}$$

Issues with resolving (dead)locks through Catalysers

Value-passing:

$$a(x).\,\text{if } \mathit{condition}(x) \text{ then } P \text{ else } Q$$

Issues with resolving (dead)locks through Catalysers

Value-passing:

$$a(x).\,\text{if } \mathit{condition}(x) \text{ then } P \text{ else } Q$$

Linearity:

$$a.\bar{a}.\mathbf{0}$$

Issues with resolving (dead)locks through Catalysers

Value-passing:

$$a(x).\,\text{if } \mathit{condition}(x) \text{ then } P \text{ else } Q$$

Linearity:

$$a.\bar{a}.\mathbf{0} \parallel \bar{a}.a.\mathbf{0} \quad \longleftarrow \quad \text{illegal!!}$$

(Dead)lock resolution by Disentangling

Example

Consider the program

$$a(x).\Big(\bar{a}\langle 1\rangle.\text{if } x \geq 1 \text{ then } \bar{b}\langle 4\rangle.\mathbf{0} \text{ else } \bar{b}\langle 2\rangle.\mathbf{0}\Big)$$

(Dead)lock resolution by Disentangling

Example

Consider the program

$$a(x).\Big(\bar{a}\langle 1\rangle.\text{if } x \geq 1 \text{ then } \bar{b}\langle 4\rangle.\mathbf{0} \text{ else } \bar{b}\langle 2\rangle.\mathbf{0}\Big)$$

What we need to do is identify the source of the (dead) lock

$$a(x).\Big(\bar{a}\langle 1\rangle.\text{if } x \geq 1 \text{ then } \bar{b}\langle 4\rangle.\mathbf{0} \text{ else } \bar{b}\langle 2\rangle.\mathbf{0}\Big)$$

(Dead)lock resolution by Disentangling

Example

Consider the program

$$a(x).\Big(\bar{a}\langle 1\rangle.\text{if } x \geq 1 \text{ then } \bar{b}\langle 4\rangle.\mathbf{0} \text{ else } \bar{b}\langle 2\rangle.\mathbf{0}\Big)$$

What we need to do is identify the source of the (dead) lock

$$a(x).\Big(\bar{a}\langle 1\rangle.\text{if } x \geq 1 \text{ then } \bar{b}\langle 4\rangle.\mathbf{0} \text{ else } \bar{b}\langle 2\rangle.\mathbf{0}\Big)$$

... and refactor the program accordingly

$$\bar{a}\langle 1\rangle.\mathbf{0} \parallel a(x).\Big(\text{if } x \geq 1 \text{ then } \bar{b}\langle 4\rangle.\mathbf{0} \text{ else } \bar{b}\langle 2\rangle.\mathbf{0}\Big)$$

Disentangling Locks

There are two components:

Disentangling Locks

There are two components:

1. Detection

Disentangling Locks

There are two components:

1. Detection
2. Refactoring

Detection: Identifying the target set

1. Simplify setting: processes using **linear ports only**.
2. Simplify further: **complete** processes.
3. We are left with two sets of processes:
   - The lock-free processes
   - The complete but **not** lock-free.

Detection: Characteristing the target set

Example (Not Lock-free)

- $a.\bar{b}.\mathbf{0} \parallel b.\bar{a}.\mathbf{0}$
- $a.(\bar{b}.\mathbf{0} \| c.\mathbf{0}) \parallel b.\bar{a}.\mathbf{0} \parallel \bar{c}.\mathbf{0}$

Detection: Characteristing the target set

Example (Not Lock-free)

- $a.\bar{b}.\mathbf{0} \parallel b.\bar{a}.\mathbf{0}$
- $a.(\bar{b}.\mathbf{0} \parallel c.\mathbf{0}) \parallel b.\bar{a}.\mathbf{0} \parallel \bar{c}.\mathbf{0}$
- $d.\left(a.\bar{b}.\mathbf{0} \parallel b.\bar{a}.\mathbf{0}\right) \parallel \bar{d}.\mathbf{0}$

Detection: Characteristing the target set

Example (Not Lock-free)

- $a.\bar{b}.\mathbf{0} \parallel b.\bar{a}.\mathbf{0}$
- $a.(\bar{b}.\mathbf{0} \| c.\mathbf{0}) \parallel b.\bar{a}.\mathbf{0} \parallel \bar{c}.\mathbf{0}$
- $d.\left(a.\bar{b}.\mathbf{0} \parallel b.\bar{a}.\mathbf{0}\right) \parallel \bar{d}.\mathbf{0}$
- $d.\left(a.\bar{b}.\mathbf{0} \parallel b.\bar{a}.\mathbf{0}\right) + d.\mathbf{0} \parallel \bar{d}.\mathbf{0}$

Detection: Characteristing the target set

Example (Not Lock-free)

- $a.\bar{b}.\mathbf{0} \parallel b.\bar{a}.\mathbf{0}$
- $a.(\bar{b}.\mathbf{0} \parallel c.\mathbf{0}) \parallel b.\bar{a}.\mathbf{0} \parallel \bar{c}.\mathbf{0}$
- $d.\left(a.\bar{b}.\mathbf{0} \parallel b.\bar{a}.\mathbf{0}\right) \parallel \bar{d}.\mathbf{0}$
- $d.\left(a.\bar{b}.\mathbf{0} \parallel b.\bar{a}.\mathbf{0}\right) + d.\mathbf{0} \parallel \bar{d}.\mathbf{0}$
- $\bar{d}.\left(a.(\bar{b}.\mathbf{0} \parallel c.\mathbf{0})\right) \parallel d.\left(b.\bar{a}.\mathbf{0} \parallel \bar{c}.\mathbf{0}\right)$

Detection: Characteristing the target set

$$\mathrm{cin}(a,P) \stackrel{\text{def}}{=} P \equiv \mathcal{C}[Q] \text{ and } \mathrm{in}(a,Q)$$

$$\mathrm{cout}(a,P) \stackrel{\text{def}}{=} P \equiv \mathcal{C}[Q] \text{ and } \mathrm{out}(a,Q)$$

$$\mathrm{tcmp}(P) \stackrel{\text{def}}{=} \begin{cases} (\mathrm{in}(a,P) \text{ implies } \mathrm{cout}(a,P)) \\ \quad \text{and} \quad (\mathrm{out}(a,P) \text{ implies } \mathrm{cin}(a,P)) \end{cases}$$

$$\mathrm{sl}(P) \stackrel{\text{def}}{=} \mathrm{dlock}(P) \text{ and } \mathrm{tcmp}(P)$$

$$\mathrm{psl}(P) \stackrel{\text{def}}{=} \text{Exists } \mathcal{E}[Q] \text{ such that } P \longrightarrow^* \mathcal{E}[Q] \text{ and } \mathrm{sl}(Q)$$

Detection: Characteristing the target set

$$\mathrm{cin}(a,P) \stackrel{\text{def}}{=} P \equiv \mathcal{C}[Q] \text{ and } \mathrm{in}(a,Q)$$

$$\mathrm{cout}(a,P) \stackrel{\text{def}}{=} P \equiv \mathcal{C}[Q] \text{ and } \mathrm{out}(a,Q)$$

$$\mathrm{tcmp}(P) \stackrel{\text{def}}{=} \begin{cases} (\mathrm{in}(a,P) \text{ implies } \mathrm{cout}(a,P)) \\ \quad \text{and} \quad (\mathrm{out}(a,P) \text{ implies } \mathrm{cin}(a,P)) \end{cases}$$

$$\mathrm{sl}(P) \stackrel{\text{def}}{=} \mathrm{dlock}(P) \text{ and } \mathrm{tcmp}(P)$$

$$\mathrm{psl}(P) \stackrel{\text{def}}{=} \text{Exists } \mathcal{E}[Q] \text{ such that } P \longrightarrow^* \mathcal{E}[Q] \text{ and } \mathrm{sl}(Q)$$

Detection: Characteristing the target set

$$\mathrm{cin}(a,P) \stackrel{\text{def}}{=} P \equiv \mathcal{C}[Q] \text{ and } \mathrm{in}(a,Q)$$

$$\mathrm{cout}(a,P) \stackrel{\text{def}}{=} P \equiv \mathcal{C}[Q] \text{ and } \mathrm{out}(a,Q)$$

$$\mathrm{tcmp}(P) \stackrel{\text{def}}{=} \begin{cases} (\mathrm{in}(a,P) \text{ implies } \mathrm{cout}(a,P)) \\ \quad \text{and} \quad (\mathrm{out}(a,P) \text{ implies } \mathrm{cin}(a,P)) \end{cases}$$

$$\mathrm{sl}(P) \stackrel{\text{def}}{=} \mathrm{dlock}(P) \text{ and } \mathrm{tcmp}(P)$$

$$\mathrm{psl}(P) \stackrel{\text{def}}{=} \text{Exists } \mathcal{E}[Q] \text{ such that } P \longrightarrow^* \mathcal{E}[Q] \text{ and } \mathrm{sl}(Q)$$

Detection: Characteristing the target set

$$\begin{aligned}
\mathrm{cin}(a,P) &\stackrel{\text{def}}{=} P \equiv \mathcal{C}[Q] \text{ and } \mathrm{in}(a,Q) \\
\mathrm{cout}(a,P) &\stackrel{\text{def}}{=} P \equiv \mathcal{C}[Q] \text{ and } \mathrm{out}(a,Q) \\
\mathrm{tcmp}(P) &\stackrel{\text{def}}{=} \begin{cases} (\mathrm{in}(a,P) \text{ implies } \mathrm{cout}(a,P)) \\ \quad \text{and} \quad (\mathrm{out}(a,P) \text{ implies } \mathrm{cin}(a,P)) \end{cases} \\
\mathrm{sl}(P) &\stackrel{\text{def}}{=} \mathrm{dlock}(P) \text{ and } \mathrm{tcmp}(P) \\
\mathrm{psl}(P) &\stackrel{\text{def}}{=} \text{Exists } \mathcal{E}[Q] \text{ such that } P \longrightarrow^* \mathcal{E}[Q] \text{ and } \mathrm{sl}(Q)
\end{aligned}$$

Example

$$a.(\bar{b}.\mathbf{0} \| c.\mathbf{0}) \;\|\; b.\bar{a}.\mathbf{0} \;\|\; \bar{c}.\mathbf{0}$$

Detection: Characteristing the target set

$$\begin{aligned}
\mathrm{cin}(a,P) &\stackrel{\text{def}}{=} P \equiv \mathcal{C}[Q] \text{ and } \mathrm{in}(a,Q) \\
\mathrm{cout}(a,P) &\stackrel{\text{def}}{=} P \equiv \mathcal{C}[Q] \text{ and } \mathrm{out}(a,Q) \\
\mathrm{tcmp}(P) &\stackrel{\text{def}}{=} \begin{cases} (\mathrm{in}(a,P) \text{ implies } \mathrm{cout}(a,P)) \\ \quad \text{and} \quad (\mathrm{out}(a,P) \text{ implies } \mathrm{cin}(a,P)) \end{cases} \\
\mathrm{sl}(P) &\stackrel{\text{def}}{=} \mathrm{dlock}(P) \text{ and } \mathrm{tcmp}(P) \\
\mathrm{psl}(P) &\stackrel{\text{def}}{=} \text{Exists } \mathcal{E}[Q] \text{ such that } P \longrightarrow^* \mathcal{E}[Q] \text{ and } \mathrm{sl}(Q)
\end{aligned}$$

Example

$$a.(\bar{b}.\mathbf{0} \| c.\mathbf{0}) \;\|\; b.\bar{a}.\mathbf{0} \;\|\; \bar{c}.\mathbf{0}$$

Detection: Other issues

Detection: Other issues

1. Compositionality:

$$\bar{d}.\Big(a.(\bar{b}.\mathbf{0} \| c.\mathbf{0})\Big) \;\|\; d.\Big(b.\bar{a}.\mathbf{0} \;\|\; \bar{c}.\mathbf{0}\Big)$$

Detection: Other issues

1. Compositionality:

$$\bar{d}.\Big(a.(\bar{b}.\mathbf{0} \| c.\mathbf{0})\Big) \;\|\; d.\Big(b.\bar{a}.\mathbf{0} \;\|\; \bar{c}.\mathbf{0}\Big)$$

Detection: Other issues

1. Compositionality:

$$\bar{d}.\Big(a.(\bar{b}.\mathbf{0} \| c.\mathbf{0})\Big) \;\|\; d.\Big(b.\bar{a}.\mathbf{0} \;\|\; \bar{c}.\mathbf{0}\Big)$$

2. Approximation:

Under or Over Approximate?

Refactoring

Obvious Criteria (and less obvious ones)

- Preserve some aspects satisfied by the entangled process.
- Improve on others.
- But there may be other criteria...

Refactoring

Obvious Criteria (and less obvious ones)

- Preserve some aspects satisfied by the entangled process.
- Improve on others.
- But there may be other criteria...

Example

Consider a detected entangled process: $a.\bar{b}.c.\mathbf{0} \parallel \bar{c}.b.\bar{a}.\mathbf{0}$

Refactoring

Obvious Criteria (and less obvious ones)

- Preserve some aspects satisfied by the entangled process.
- Improve on others.
- But there may be other criteria...

Example

Consider a detected entangled process: $a.\bar{b}.c.\mathbf{0} \parallel \bar{c}.b.\bar{a}.\mathbf{0}$
We can disentangle it as: $(a.\mathbf{0} \parallel \bar{b}.c.\mathbf{0}) \parallel (\bar{c}.\mathbf{0} \parallel b.\bar{a}.\mathbf{0})$

Refactoring

Obvious Criteria (and less obvious ones)

- Preserve some aspects satisfied by the entangled process.
- Improve on others.
- But there may be other criteria...

Example

Consider a detected entangled process: $a.\bar{b}.c.\mathbf{0} \parallel \bar{c}.b.\bar{a}.\mathbf{0}$
We can disentangle it as: $(a.\mathbf{0} \parallel \bar{b}.c.\mathbf{0}) \parallel (\bar{c}.\mathbf{0} \parallel b.\bar{a}.\mathbf{0})$
Or else disentangle it as: $(\bar{a}.\mathbf{0} \parallel a.\bar{b}.c.\mathbf{0}) \parallel (\bar{c}.\mathbf{0} \parallel b.\mathbf{0})$

Value Passing and Static Binding

Example

$$a(x).\bar{b}\langle x+1\rangle.c(y).\mathbf{0} \;\parallel\; \bar{c}\langle 5\rangle.b(z).\bar{a}\langle 7\rangle.\mathbf{0}$$

Value Passing and Static Binding

Example

$$a(x).\bar{b}\langle x+1\rangle.c(y).\mathbf{0} \;\parallel\; \bar{c}\langle 5\rangle.b(z).\bar{a}\langle 7\rangle.\mathbf{0}$$

Value Passing and Static Binding

Example

$$a(x).\bar{b}\langle x+1\rangle.c(y).\mathbf{0} \;\|\; \bar{c}\langle 5\rangle.b(z).\bar{a}\langle 7\rangle.\mathbf{0}$$

You can disentangle it as:

$$a(x).\bar{b}\langle x+1\rangle.c(y).\mathbf{0} \;\|\; \bar{c}\langle 5\rangle.\mathbf{0} \;\|\; b(z).\mathbf{0} \;\|\; \bar{a}\langle 7\rangle.\mathbf{0}$$

Value Passing and Static Binding

Example

$$a(x).\bar{b}\langle x+1\rangle.c(y).\mathbf{0} \;\|\; \bar{c}\langle 5\rangle.b(z).\bar{a}\langle 7\rangle.\mathbf{0}$$

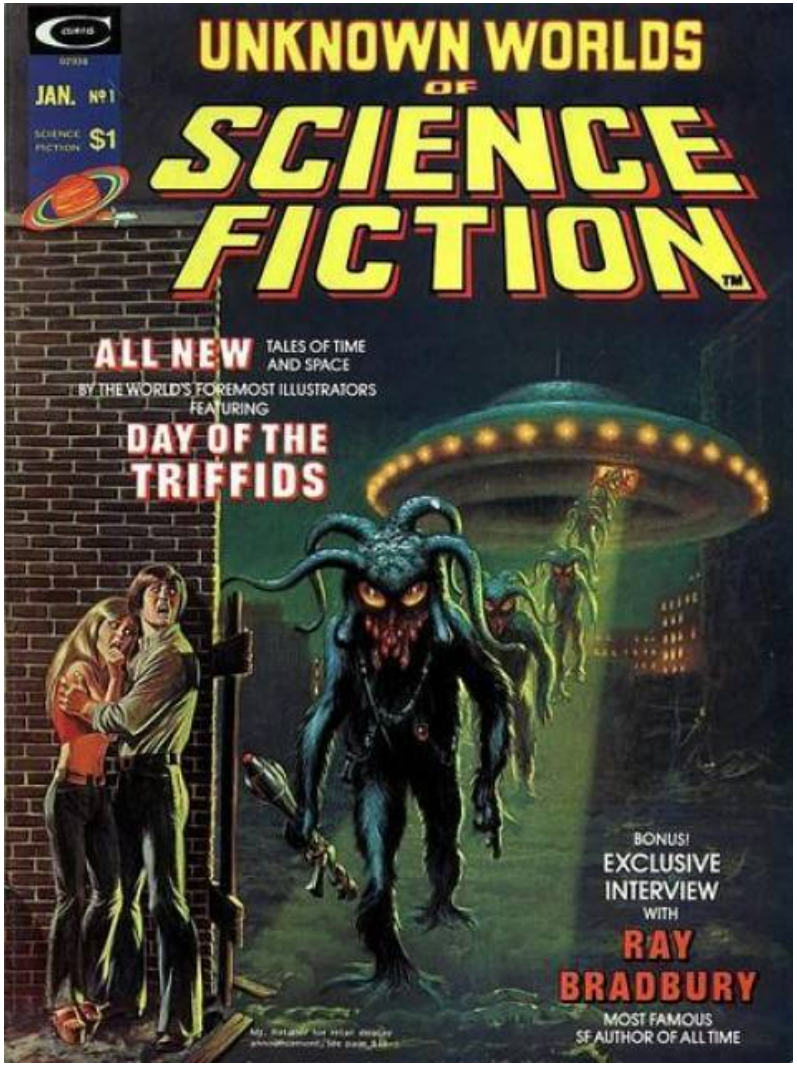
You can disentangle it as:

$$a(x).\bar{b}\langle x+1\rangle.c(y).\mathbf{0} \;\|\; \bar{c}\langle 5\rangle.\mathbf{0} \;\|\; b(z).\mathbf{0} \;\|\; \bar{a}\langle 7\rangle.\mathbf{0}$$

But what about:

$$a(x).\bar{b}\langle x+1\rangle.c(y).\mathbf{0} \;\|\; \bar{c}\langle 5\rangle.b(z).\bar{a}\langle z-1\rangle.\mathbf{0}$$

Channel Passing



Conclusion

- The problem of disentangling seems to be quite different from its dual counterpart, typechecking (dead)lock-free processes.
- We can characterise the dual of lock-free processes as potentially self-locking processes.
- There is a tension when detecting locks between compositionality and appoximation.
- The criteria for adequate refactoring are not obvious and some may be conflicting.