The University of Oxford

MSc (Mathematics and Foundations of Computer Science)

Concurrency

Trinity Term 2020

This mini project will be marked out of 100, with marks allocated as shown to the parts set out below.

This mini project is about the CSP language and behaviourally-based equivalences over it. The core CSP language, which you should consider, consists of a finite alphabet Σ (which has at least 2 members, and which where required can be finitely extended) and the constant and operators $STOP$, prefix, internal $\sqcap$ and external $\Box$ choice, hiding $\setminus A$ and interface parallel $\underset{A}{\|}$, renaming $P[[R]]$ under relations that are total for the events of P, the throw operator Θ_A and recursion. There is no need to consider $SKIP$ or sequential composition. Other operators are definable in terms of this core. We will add the *priority* operator in the final part.

Part 1. How, in this language, can you model DIV, the simply divergent process, alphabet parallel ${}_A\|_B$ and interleaving $|||$? **10 marks**

Part 2. Consider the following "models" of CSP: each is an equivalence on CSP terms or nodes in an LTS determined by class(es) of observations that can be made.

A The *singleton failures model* represents a process by its finite traces T, and by the set of pairs (s, a), where s is in T and a is an event it can stably refuse after s.

B The *reduced stable failures model* represents a process by only its stable failures (s, X), where X is a set of events it can refuse after s.

C The *extended stable revivals model* consists of the set T of all finite traces, plus all triples of the form (s, X, t), where (s, X) is a stable failure and t is a trace it can go ahead and perform from a state that refuses X after s.

In each case $\mathcal{M}$ quote a pair of processes P and Q (as terms in the core CSP language above) that are distinguished by one of the stable failures model $\mathcal{F}$ and $\mathcal{M}$, but not the other. **15 marks**

Part 3. Order these three equivalences and the usual traces and stable failures models (so five equivalences in all) in terms what distinctions it makes: one model $\mathcal{M}$ is more refined than a second one $\mathcal{N}$ if every pair of processes identified by $\mathcal{M}$ is identified by $\mathcal{N}$. Which of them is strong enough to distinguish

(i) if a process can deadlock, and

(ii) if a divergence free process is deterministic?

Give reasons for your answers. **15 marks**

Part 4. One of the three equivalences described in Part 2 is compositional for the whole language set out above, and the other two are not. Which are these? In the non-compositional cases, identify the operator or operators where compositionality fails, giving examples. In the compositional case, give the semantics of hiding. **15 marks**

Part 5. Find a context $C[p]$ (i.e. a process term with a process variable p) such that P is refined by Q over reduced stable failures if and only if $C[P]$ is refined by $C[Q]$ over stable failures. Hint: clearly $C[P]$ and $C[Q]$ are stable failures equivalent whenever P and Q have no stable states, so $C[\cdot]$ must map them to the same value. **10 marks**

Part 6. The *priority* operator is added to CSP as follows. It has a parameter which is a partial order $\leq$ that prioritises the events Σ, together with the invisible τ action. τ is always maximal under $\leq$, though not necessarily maximum as other maximal events $a \in \Sigma$ are allowed to be incomparable to τ. Its operational semantics are described.

$$\frac{P \xrightarrow{a} P' \wedge \forall\, b \neq a.a \leq b \Rightarrow b \notin \mathit{initials}(P)}{\mathbf{prioritise}(P, \leq) \xrightarrow{a} \mathbf{prioritise}(P', \leq)}\ (a \in \Sigma \cup \{\tau\}).$$

where $\mathit{init}(P)$ are the initial actions in the operational semantics of P, including possibly τ.

This operator is described in Section 20.2 of *Understanding Concurrent Systems*. Priority fails to be compositional over many CSP models, including traces, stable failures and the three above.

Thanks to the technique known as *model shifting*, making crucial use of priority, refinement over a large class of models (including all this question considers) can be reduced to trace refinement. Study the paper *Translating between models of concurrency* (Mestel and Roscoe), downloadable from
`https://link.springer.com/article/10.1007/s00236-020-00372-9`
up to Section 3.2 which shows how stable failures refinement is reduced to trace refinement.

For each of singleton failures and extended revivals, give an analogous construction to that in Section 3.2 in the paper, for reducing its refinement relation to trace refinement. **35 marks**