Lição 1: Contexto, Objectivos e Plano

Luís Soares Barbosa

Sumário

Esta Lição introdutória procura motivar os alunos para o curso, definindo com rigor, mas de forma necessariamente breve, os seus objectivos e enquadrando a matéria que se propõe abordar no contexto mais amplo dos modelos e cálculos para sistemas computacionais. De seguida introduz o programa do curso, sob a forma de um conjunto de questões as quais, no final, o aluno é suposto saber problematizar. Por fim é apresentada a bibliografia recomendada, o método a seguir e o sistema de avaliação.

Disciplina: Métodos de Programação IV (2005-06)
Docente *Luís Soares Barbosa,* Departamento de Informática, Universidade do Minho

---

1 Enquadramento e Objectivos do Curso

Onde queremos chegar?

Este curso procura fornecer uma base conceptual sólida para a compreensão da computação reactiva. A partir dela o curso assume como seu principal objectivo o desenvolvimento das capacidades de modelação rigorosa e raciocínio sobre colecções arbitrárias de sistemas computacionais que evoluem concorrentemente e interactuam ao longo dessa evolução.

Este tipo de sistemas é-nos tecnologicamente familiar. Poderíamos mesmo dizer que não só sustentam boa parte do tecido económico-social em que as nossas sociedades se movem, como são pervasivos do nosso quotidiano pessoal. Do sistema de controlo da temperatura de um edifício à coordenação dos sistemas bancários ou médicos a nível de um país, dos protocolos que asseguram as comunicações numa rede local à Internet com todos os componentes móveis que itermitentemente se ligam e desligam às diveras plataformas de software que sobre ela emergem, a lista de exemplos parece não ter fim. Interacção, concorrência, mobilidade, distribuição, ubiquidade, computação global, são palavras-chave no entendimento destes sistemas. A verdade, porém, é que quanto mais dependentes nos tornamos destes sistemas, maior a necessidade que temos de os entender e, como informáticos, de ser capazes de os conceber, analisar e certificar.

É neste ponto que se torna crucial o papel dos modelos matemáticos — abstracções que captam com rigor as características emergentes de um dado paradigma de computação para permitir analisar e prever as suas propriedades. Trata-se, evidentemente, de um papel comum a toda a Ciência, mas incontornável em Ciências da Computação na medida em que o comportamento exernamente manifesto de um sistema computacional é o seu único aspecto que é possível observar de forma directa.

Os modelos, enquanto objectos formais, possibilitam o raciocínio rigoroso sobre os sistemas que representam. Talvez o primeiro, e mais célebre, testemunho do sucesso desta aproximação em Ciências da

Computação tenha sido a solução negativa dada por A. Turing em [Tur37] ao *Entscheidungsproblem* de Hilbert, mostrando que o problema de determinar a terminação de uma máquina de Turing executando sobre um dado de entrada era algoritmicamente impossível.

Raciocinar no interior (e a partir) de um modelo, faz-se através de *cálculos*, e essa é a própria matéria da Matemática enquanto, na feliz expressão de R. Backhouse, *art of effective reasoning* [Bac01]. Por isso modelação e cálculo constituem os pilares de uma efectiva disciplina de Engenharia. Mesmo num domínio, como a Informática, que se popularizou como tecnologia muito antes de se haver constituido como ciência.

De onde partimos?

Mas de que modelos dispomos para a computação?

Uma das abstracções mais populares para descrever o significado de um programa é a noção matemática de *função*, i.é uma regra de cálculo que transforma a informação fornecida em nova informação que constitui a resposta ao problema proposto. Essa resposta é duplamente definida: de cada vez que o cálculo se repete é produzida uma resposta e esta é invariavelmente a mesma.

Esta associação de um significado funcional ao processo computacional é típica não apenas das linguagens de programação funcional, que ocuparam um lugar de destaque em disciplinas anteriores, mas igualmente bem sucedida em outros sistemas cujo comportamento se pode descrever em termos de uma *transformação* de valores. Sistemas que se preocupam essencialmente com a *estrutura* e a *transformação* da informação. Por exemplo, podemos modelar uma rotina escrita numa linguagem imperativa por uma função cujo argumento e resultado é de um tipo que modela a noção de memória endereçável (onde reside o estado sobre o qual a rotina actua).

O modelo funcional é muito mais rico do que, à partida, poderíamos suspeitar. De facto, surge dotado de um cálculo clássico, preciso e sóbrio — o λ-*calculus* [Bar80] e sugere, pela sua própria natureza, uma semântica formal para a programação[1], que é estritamente modular, na medida em que a composição de programas é interpretada como composição de funções. Foi a partir deste modelo que se desenvolveram ramos fundamentais das ciências da computação, nomeadamente a *teoria dos tipos* (com toda a diversidade de conceitos como funções de ordem superior, polimorfismo, subtipos, etc.), a *semântica denotacional* e, por fim, o estudo conjunto da *complexidade* e *análise dos algoritmos*.

Neste modelo, porém, o objecto da computação é essencialmente associado à *transformação de informação*: cada problema é especificado pela colecção dos seus dados admissíveis e, para cada um deles, o resultado esperado. Esta transformação não é necessariamente total (a parcialidade é necessária para exprimir situações de indefinição ou não terminação), nem sequer estritamente funcional (a modelação por relações permite explorar o não determinismo inerente a diversas situações).

Nota 1 **[Os Primeiros Modelos]**

O estudo de modelos para a computação é anterior à construção dos primeiros computadores. Uma das questões fundamentais na década de 30 dizia respeito ao conceito de computabilidade [Kle35, Chu36]— um resultado chave dessa época foi a identificação da classe das funções ditas computáveis como aquelas que eram exprimíveis no λ-*calculus*, ou, equivalentemente, programáveis numa máquina de Turing. Para além da computabilidade, as máquinas de Turing forneceram uma base para o estudo das classes de complexidade dos algoritmos, enquanto o λ-*calculus* [Chu41] se tornou fundamental no desenvolvimento de todo um paradigma de programação e na génese da teoria de tipos.

Os anos 50 assistem a um surto de investigação em torno das linguagens formais, em boa parte motivada pela tentativa de compreender e modelar a inteligência humana de forma computacional. Kleene [Kle56] provou que as linguagens regulares eram reconhecidas por um modelo computacional particularmente importante: os autómatos de estado finito. Mais tarde Rabin e Scott mostraram que adicionando não determinismo ao modelo não se incrementava o seu poder expressivo. Em 1956, um artigo fundamental de Chomsky [Cho56] introduz uma hierarquia de gramáticas para a descrição de liguagens formais. Hierarquia

[1] Programas são funções, ou mais genericamente, homomorfismos, entre conjuntos, eventualmente com alguma estrutura adicional, que representam os tipos da informação envolvida.

que abarca desde as lingugens regulares acima referidas, às linguagens independentes do contexto, modeladas pelos *push-down automata*, em que hoje se baseia a maioria das linguagens de programação, e às linguagens recursivamente enumeráveis que são reconhecidas, mas não decididas, pelas máquinas de Turing.

Por essa altura, no entanto, no lado tecnológico, apareciam os primeiros sistemas operativos e, um pouco mais tarde, nos anos '60, os sistemas de multiprogramação. Com eles os primeiros problemas levandatos pela execução concorrente e a interferência entre diferentes cursos de execução. A teoria começava a ficar para trás na sua capacidade de explicar os desenvolvimentos técnicos. A computação reactiva emergia com uma relevância porventura insuspeita.

Neste modelo, como vimos, o objecto da computação está essencialmente associado à *transformação de informação*: cada problema é especificado pela colecção dos seus dados admissíveis e, para cada um deles, o resultado esperado. Esta transformação não é necessariamente total (a parcialidade é necessária para exprimir situações de indefinição ou não terminação), nem sequer estritamente funcional (a modelação por relações permite explorar o não determinismo inerente a diversas situações).

Que tipo de sistemas nos interessa?

Apesar do seu sucesso no entendimento da natureza da programação, que permitiu o desenvolvimento notável de linguagens, sistemas e tecnologias, é consensual reconhecer ao modelo funcional da computação descrito acima uma aplicabilidade limitada. A maior parte dos sitemas referidos no início desta Lição apresentam características dificilmente enquadráveis neste modelo:

- Antes de mais são sistemas que interagem com o seu ambiente de forma continuada (alguns mesmo de forma quase permanente), *ao longo* da sua execução, e não apenas no *início* e no *fim* desta. Reagem a estímulos, em função dos quais alteram eventualmente o seu comportamento. Ao mesmo tempo influenciam o seu ambiente trocando informação com ele ou inciando operações que o afectam.
- São inerentemente paralelos, executando de forma concorrente e colaborando ou competindo através da interacção mútua. O que levanta um conjunto muito prórpio de problemas. A presença de comportamento *concorrente* começa logo por comprometer a composicionalidade de que anteriormente nos vangloriamos. E isto porque programas activados em paralelo podem *interferir* uns com os outros, de uma forma que, de resto, será, em muitos casos, não determinística. Pense-se, por exemplo, na activação paralela dos programas $P_1 =$ `x:=1; x:=x+1` e $P_2 =$ `x:=2`, mesmo tomando as atribuições como acções indivisíveis (i.é, que não podem ser interrompidas).
- A sua análise e construção requer que o não determinismo seja tomado como uma noção elementar incontornável, e não como um artífício técnico que acomoda estados temporários instáveis até que todos os valores do problema sejam fornecidos. Na maior parte dos casos nunca o serão.
- Tipicamente o seu ciclo de vida não é limitado: a não terminação não é uma excepção mas uma característica altamente desejável. Como nota Reisig, em [Rei88], neste tipo de sistemas *initial states do in general not exist and terminal states might denote disaster rather than result*.
- Organizam-se, não raro, em redes por onde se dissemina informação e capacidade de processamento, com grande heterogeneidade e ambientes muitas vezes voláteis e critícos em termos de segurança e confiabilidade, relevando para todo um conjunto de fenómonos e problemas, muitas vezes referidos sob a metáfora da *computação global* que estamos ainda longe de entender em todas as suas manifestações e complexidade.
- Interligam-se, não raro, de forma efémera, movendo-se de um ponto para outro de um espaço computacional global ou alterando, em tempo de execução, a estrutura das suas ligações.

De uma forma geral classificaremos estes sistemas como *reactivos*, seguindo um termo proposto por David Harel e Amir Pnueli em [HP85]. Outros qualificativos, como *concorrente*, *móvel*, *distribuído* ou

interactivo, captam apenas elementos parcelares na sua caracterização, ou possuem, como no último caso, significados mais específicos em Ciências da Computação.

Se, porém, tivessemos de escolher um conceito chave para a sua caracterização, a noção de *interacção* (sustentada ao longo do tempo e dinamicamente estabelecida) seria, sem dúvida, escolhida. Reparemos que o pequeno exemplo acima referido identifica um (senão **o**) ingrediente fundamental nos sistemas que nos propomos estudar: a *interacção* entre os processos activados concorrentemente[2]. Quando a interacção é de natureza sequencial, encontra representação na estrutura dos termos funcionais e o λ-*calculus* — enquanto teoria das funções recursivas como regras de transformação e do seu comportamento aplicativo — fornece o modelo adequado. No caso geral, contudo, será necessário enriquecer o cálculo, dotando-o de um conjunto de identificadores de *portas* usadas para enviar/aceitar valores, eventualmente em simultâneo. No exemplo dado a interação entre os dois programas resumia-se a uma interferência resultante da manipulação de uma variável partilhada. Outras formas de interacção mais complexas ocorrem na prática. Note-se, de passagem, que a semelhança entre a expressão-λ $\lambda x.P$ e a expressão $\alpha_x.P$, que, na linguagem que vamos introduzir significa a prefixação do processo P pela acção de receber um valor na porta α, sugere λ como (o) identificador de uma porta de interação do termo, que é única, precisamente na medida em que a avaliação é sequencial.

Um modelo matemático para sistemas reactivos deverá, pois, possibilitar a descrição do comportamento de conjuntos de sistemas evoluindo concorrencialmente e interagindo uns com os outros. Deve, também, permitir exprimir fenómonos associados à concorrência e que nos são familiares a partir, por exemplo, do estudo dos sistemas operativos, das redes ou da computação paralela em geral: *e.g.*, *deadlock*, *livelock*, exclusão mútua, etc.

Outro ingrediente a ter em conta relaciona-se com o lugar que a *observação* tem na análise destes sistemas. No modelo funcional, e suas generalizações, estamos sobremaneira preocupados com a estrutura da informação em jogo e as suas transformações. Aqui preocupar-nos-á sobretudo o elemento dual, i.é a observação dos comportamentos dos sistemas e das suas interacções. Nas palavras de Robin Milner a ciência da computação tornou-se uma *teoria estrutural da interacção*: *Thus software, from being a prescription for how to do something — in Turing's terms a "list of instructions" — becomes much more akin to a description of behaviour, not only programmed on a computer, but occurring by hap or design inside or outside it* [Mil97].

Tal será o objecto de estudo neste curso. Iremos prossegui-lo a partir de um conjunto de questões que enunciaremos a seguir e adoptando uma abordagem — baseada em *álgebras de processos* — que constituindo um corpo de conhecimentos sólido e com uma influência importante na prática (veja-se, por exemplo, o desenvolvimento recente do Cω [BCF02] pela Microsoft), não se pretende canónica nem é certamente única.

Porcuraremos, ao longo deste curso, desenvolver paralelamente as componentes de *modelação* e *cálculo* — os dois pilares gémeos que, no espírito do Grupo Disciplinar de Lógica e Métodos Formais em que esta disciplina se integra, constituem a base metodológica da Engenharia de Software.

Nota 2 **[Modelos para a Computação Reactiva: Interacção & Concorrência]**

A interligação, por vezes subtil, das noções de *interacção* e *concorrência* está na génese de um conjunto de modelos para a computação reactiva que seria impossível rever, ou sequer enunciar, nesta Lição introdutória — o aluno mais destemido é referido a [WN95] para uma classificação e apresentação de alguns dos marcos mais importantes nesta paisagem. Essa investigação lançou também as bases para o desenvolvimento de ferramentas (semi-)automáticas de verificação de sistemas reactivos, com um impacto aplicacional relevante.

Anotaremos, aqui, que foi provavelmente Carl-Adam Petri o primeiro a chamar a atenção, no início dos anos 60 (em [Pet62]), para a necessidade de novos paradigmas para descrever e raciocinar sobre sistemas que interagem concorrencialmente. A teoria das redes popularizadas com o seu nome baseia-se precisamente na possibilidade de uma transição num sistema depender de (i.é, interagir com) outra que ocorre noutro sistema que evolui concorrencialmente com o primeiro.

Tal como as redes de Petri, a grande maioria dos modelos para a computação reactiva evoluiram a partir da noção de *automato*, da qual se retém essencialmente a estrutura de *transições etiquetadas* — precisamente pelas interacções em que o sistema se envolve.

[2] De facto dois computadores distintos e activos, sem qualquer ligação entre si, constituem um sistema concorrente que não justificaria as 52 horas lectivas previstas para este curso...

Tais sistemas, ditos *de transição*, exibem comportamentos — usualmente designados por *processos* — que podem ser descritos por linguagens formais e que, mais importante ainda, formam domínios nos quais se podem definir (ou identificar) determinadas estruturas algébricas. I.e., operadores que os combinam e que exibem um conjunto suficientemente rico de propriedades que permitem combinar os padrões de interacção dos vários sistemas em presença. É esta a área de um conjunto de modelos que exploram a estrutura algébrica dos comportamentos de sistemas reactivos e que se popularizou, a partir de finais dos anos '70 sob a designação de álgebras de processos. Adoptando uma noção muito primitiva de interacção — a de comunicação síncrona por participação numa interacção comum — este tipo de modelos tem em CCS (acrónimo de *calculus of communicating systems* [Mil80]) o representante paradigmático. Estão também na base de CSP (acrónimo de *communicating sequential processes*) uma proposta de C. A. R. Hoare para uma linguagem de programas paralelos [Hoa78] com um enorme impacto na área, assim como de um sem número de cálculos e modelos que adoptam princípios similares (cf, por exemplo, [Hoa85, Hen88, BK85], para citar alguns dos de maior impacto).

Do ponto de vista tecnológico, a noção de uma comunidade de sistemas interactuantes, dotados de identidade própria e persistência no tempo, mas também de capacidade de evolução por alteração do seu estado, emerge na noção de *objecto*, que tem raízes nas linguagens de simulação muito em voga nos anos '60, e em particular em SIMULA [DMN68]. Os anos '90 assitiram ao nascimento da Internet, que promoveu concorrência e distribuição a patamares antes não entrevistos e introduziu toda uma família de novos problemas e desafios ao nível da reconfigurabilidade das conexões, segurança e confiabilidade, mobilidade das componentes, incerteza e ausência de controlo centralizado. Em 1995 é introduzida a linguagem JAVA facilitando e disseminando a programação de processos concorrentes (*threads*) e a construção de código movél (na forma de *applets*). A emergência da *computação orientada ao serviço* [PG03, Fia04] representa uma nova e excitante arena de desafios, com o adicional *charm of inventing the science of navigation while already on board ship* [Mil97].

Na teoria o final dos anos '80 assiste à introdução do π-*calculus* [MPW92] no duplo papel de teoria dos sistemas móveis e de modelo geral para a computação que elege a interacção como noção primitiva. A influência do π-*calculus*, na investigação em álgebras de processos, nos últimos 15 anos foi enorme. Desenvolvimentos subsequentes, e que não teremos oportunidade de abordar neste curso, incluem os cálculos com representações espaciais explícitas (por exemplo, [CG98]), a incorporação de dimensões de segurança e confiabilidade (por exemplo, [AG99]), os chamados modelos construtivos da interacção (por exemplo, [Arb04]), entre muitos outros.

---

2 Programa

As Questões

Antes de enumerarmos a meia dúzia de tópicos que constituem o Programa deste curso, procuremos introduzi-los através de um conjunto de questões que orientarão o nosso esforço comum no curso. Na subsecção seguinte poderá, então, encontrar o programa formal da disciplina. Assim,

1. *Como modelar sistemas reactivos e estudar a sua composicionalidade?* Começaremos por caracterizar sistemas reactivos a partir de dois modelos já estudados em disciplinas anteriores: as funções sobre estruturas de dados infinitas e os autómatos. Chegaremos, assim, à noção de sistema de transição etiquetado sobre o qual desenvolveremos duas ferramentas básicas de análise: as relações de simulação e bissimulação. Introduziremos depois uma noção de comportamento, ou processo, enquanto padrão de evolução das capacidades de interacção de um sistema, uma linguagem formal para os descrever, cuja semântica operacional estudaremos. A noção básica é a de transição etiquetada e, como veremos, o modelo operacional da nossa linguagem emerge como uma generalização da noção de *autómato*. Basicamente, consideraremos autómatos não determinísticos, sem estados finais e sobre os quais é possível definir noções de observação mais elaboradas do que a clássica *linguagem aceite*. É curioso como, tendo os autómatos constituido um dos primeiros modelos matemáticos da computação, estão na gênese de dois conceitos base das ciências da computação: o de *estrutura de dados* (em que o espaço de estados do autómato é codificado numa álgebra) e o de *processo*, num sentido que procuraremos tornar claro ao longo deste curso.
2. *Em que condições podemos dizer que dois sistemas exibem o mesmo comportamento?* Uma definição de equivalência entre sistemas dinâmicos é extremamente útil na prática. Suponha-se, por exemplo, que necessitamos de substituir uma determinada componente de um sistema e que a componente original está esgotada ou se tornou muito cara. Será seguro substitui-la por outra? Intuitivamente diremos que a substituição será segura se o comportamento do sistema permanecer essencialmente

inalterado. Mas o que é que isso significa? Uma definição formal de equivalência fornece-nos a resposta esperada e estabelece formas de provar que a substituição é segura. Neste sentido iremos introduzir algumas noções de equivalência entre processos, baseadas no conceito de *bissimulação*, i. é, na capacidade de dois processos se "imitarem" mutuamente ao longo de toda a sua evolução. Veremos como diversos resultados genéricos de equivalência podem ser captados na forma de equações, o conjunto das quais constitui um poderoso *cálculo* (equacional) que nos ajudará a classificar, simplificar e, em geral, raciocinar sobre processos[3].

3. *De que forma é possível captar sistemas reactivos cuja estrutura de interacções se altera dinamicamente ao longo da computação?* Neste ponto o *calculus of communicating systems*, CCS [Mil80, Mil89], estudado previamente, dará lugar ao π*-calculus* [MPW92], uma profunda generalização deste que permite exprimir *mobilidade* e reconfiguração dinâmica das ligações entre processos. Veremos, em particular, como identificadores de ligações podem ser criados dinamicamente e comunicados através de outras interacções para uso posterior pelos processos receptores. Estudaremos, de novo, aspectos relacionados com a modelação, equivalência e cálculo destes sistemas.

4. *Como formular propriedades sobre sistemas reactivos (e que propriedades são, de facto, relevantes?) e como raciocinar sobre elas?* Serão abordadas algumas lógicas *modais* (i.é, em que a noção de verdade é relativa ao *modo* como a realidade é percebida, tipicamente, no nosso caso, ao estado de evolução do processo) e *temporais* (i.é, em que a verdade é indexada às computações). Procuraremos caracterizar propriedades locais e temporais e introduzir uma breve taxonomia das propriedades típicas dos sistemas concorrentes (cuja implementação corresponde, de forma igualmente típica, a problemas importantes de engenharia). Este tópico consistirá essencialmente numa introdução à lógica de Hennessy-Milner [HM85] e ao μ*-calculus* modal [Koz83].

5. *Como analisar e simular modelos de sistemas reactivos?* Neste ponto a nossa abordagem será muito pragmática, incidindo na utilização laboratorial de ferramentas computacionais para análise, teste e verificação de processos, em particular, o CONCURRENCY WORKBENCH (CWB-NC) e o MOBILITY WORKBENCH (MWB) [VM94].

O Programa

1. **Motivação:**
   Sistemas reactivos, interacção e comportamento. Modelos e cálculos em Ciências da Computação e no projecto de engenharia de software.

2. **Sistemas de transição:**
   Autómatos e tipos coindutivos. Definição de sistema de transição. Morfismos. Noção de simulação e bissimulação. Resultados básicos.

3. **Modelação de sistemas reactivos:**
   Processos e interacções. Introdução à modelação de processos em CCS. Semântica operacional. Análise e verificação de transições. Experimentação no CWB-NC.

4. **Cálculo de sistemas reactivos 1:**
   Equivalência estrita e observacional em CCS. Teorema da expansão. Teorias associadas a $\sim$, $\approx$ e $=$. Resolução de equações. Estudo de casos: sistemas de estrutura evolutiva. Experimentação no CWB-NC.

5. **Cálculo de sistemas reactivos 2:**
   Introdução ao π*-calculus*. Motivação, sintaxe, semânticas e equivalências entre processos móveis. Modelação e cálculo de processos em π*-calculus*. Estudo de casos. Experimentação no MWB.

[3] Estas "leis" têm um estatuto similar (e desempenham um papel análogo) ao, por exemplo, das leis da associatividade do produto ou da distributividade da soma sobre o produto no cálculo aritmético.

6. **Lógicas para processos:**
   Modalidade e temporalidade. Introdução à lógica de Hennessy-Milner e ao μ-*calculus* modal. Taxonomia das propriedades temporais.

3 Método e Avaliação

Método

O curso é organizado em 8 Lições que, no seu conjunto, cobrem os vários aspectos do programa. Cada Lição comporta uma componente expositiva que, conforme os casos, poderá ocupar 1 a 3 aulas teóricas e uma componente experimental que se poderá estender por igual número de aulas teorico-práticas. Esta componente será dedicada quer à resolução de exercícios, preferencialmente em grupo, quer à resolução de 3 Fichas Laboratoriais por recurso a ferramentas de análise e simulação de processos: o CWB-NC e o MWB.

Cada Lição é suportada por um guião que inclui o conjunto de notas de apoio à exposição da matéria e um conjunto de exercícios propostos. Sublinha-se que estes guiões não foram concebidos com a preocupação de formarem um texto didático independente, mas tão só como um resumo alargado das aulas correspondentes. Os alunos são encorajados a fazerem uma leitura em diagonal do guião antes de cada Lição para situarem a matéria e controlarem o volume de notas que devem tomar. Em caso algum substituem a consulta da bibliografia recomendada descrita na secção seguinte.

Cada guião é disponibilizado no `URL` do curso até uma semana antes da primeira aula da Lição respectiva.

No início do semestre será disponibilizada no `URL` do curso uma Ficha de Trabalho Individual com um conjunto de exercícios de complexidade e âmbito variável que cada aluno, de forma individual, deverá resolver até à data do exame. Esta Ficha visa ilustrar o tipo de problemas que os alunos deverão ser capazes de resolvar no final do curso. Consiste, ainda, num instrumento de auto-avaliação que permite a cada um ir medindo o seu desempenho no curso e o progressivo domínio da matéria. A Ficha inclui pelo menos um tópico que obriga a leitura de um artigo de investigação, em inglês, e sua análise critíca em diálogo com a matéria leccionada.

É convicção do docente que, num curso universitário, nada de essencial se aprende se não resultar da apropriação pessoal e do esforço de síntese, aplicação e interrogação da matéria proposta. Nesse sentido, a participação dos alunos em todos os momentos de qualquer aula é sempre valorizada, assim como o recurso ao horário de atendimento semanal do docente para esclarecimento de dúvidas e discussão de questões relacionadas com a matéria proposta no curso e sua relação com a globalidade do seu percurso universitário.

A página do curso na Internet, alojada no *wiki* do Grupo de Lógica e Métodos Formais, em

`wiki.di.uminho.pt/twiki/bin/view/Education/MetodosProgramacaoIV`

funcionará como lugar priveligiado de interacção com o docente.

Avaliação

A avaliação (formal) da disciplina é realizada através de duas componentes

- Exame global de acordo com o regulamento em vigor para as disciplinas semestrais, cuja classificação será ponderada com um factor de 0.7.

- A Ficha de Trabalho Individual, ponderada com um factor de 0.3. Apesar de os alunos serem convidados a abordar em conjunto os problemas propostos na Ficha e de o docente estar disponível para esclarecer dúvidas e discutir as resoluções propostas, sublinha-se o caracter individual deste trabalho. Cada aluno deve apresentar a sua própria resolução, escrita à mão, de forma original, concisa e clara.

A classificação obtida por avaliação formal poderá ser ajustada, em qualquer direcção, mas com uma variação não superior a dois valores, de acordo com a avaliação contínua do aluno. Esta inclui a assiduidade, participação nas aulas e a realização satisfatória das sessões laboratoriais propostas.

4 Bibliografia Recomendada

A matéria central da disciplina está amplamente detalhada nas referências seguintes que constituem a *bibliografia recomendada*. Referências mais específicas são apresentadas no guião que acompanha cada uma das Lições. Todos os livros referidos estão disponíveis na Biblioteca da Universidade. Outros textos, disponibilizados pelos seus autores na Internet, serão acessíveis através do `URL` da disciplina.

Naturalmente entre as referências sugeridas e o decurso das aulas haverá algumas (ligeiras) discrepâncias notacionais e outras (menos ligeiras) no que concerne à ordem de introdução dos assuntos e à ênfase colocada em alguns problemas ou conceitos.

[Mil99] Trata-se do livro base para as Lições 3 a 6. A primeira parte introduz o cálculo base de sistemas reactivos, popularizado sob a designação de CCS, enquanto a segunda parte aborda o problema da mobilidade e reconfiguração dinâmica de sistemas, constituindo a referência base para o *π-calculus*.

Alternativas A referência [Mil89], obra anterior do mesmo autor, é suficiente como introdução a CCS, mas omissa no que concerne à mobilidade. Uma introdução alternativa ao *π-calculus* é o livro [SW01], de caracter mais amplo e tecnicamente mais exigente, que cobre um conjunto de tópicos muito para além do exigido neste curso. Será recomendado para o aluno mais interessado a quem, contudo, se aconselha uma primeira incursão no texto [Par01], incluido como capítulo no *Handbook of Process Algebra*, do qual se disponibiliza uma versão electrónica. Em

`lamp.epfl.ch/mobility/`

poderá aceder à página mais completa sobre cálculos de processos móveis e computação global, ligações (por vezes também elas móveis!) a páginas de ferramentas, métodos, grupos de investigação e repositórios de documentação. O endereço

`www.afm.sbu.ac.uk/concurrent/`

remete-o para a *Oxford Virtual Library on Concurrent Systems*, um repositório extenso e razoavelmente catalogado sobre modelos e cálculos para sistemas reactivos e concorrentes.

[Sti92] é uma introdução às lógicas modais e temporais para especificação de processos, a abordar na Lição 7. A referência base sobre *μ-calculus* é o artigo de D. Kozen [Koz83].

Ferramentas
As duas ferramentas que serão utilizadas na disciplina — o CWB-NC e o MWB — são distribuídas livremente, com abundante documentação associada, em

`www.cs.sunysb.edu/~cwb/` e `user.it.uu.se/~victor/mwb.shtml`

respectivamente.

Referências

[AG99] M. Abadi and A. Gordon. A calculus for cryptographic protocols. *Jour. of Information and Computation*, 143:1–77, 1999.

[Arb04] F. Arbab. Reo: a channel–based coordination model for component composition. *Mathematical Structures in Comp. Sci.*, 14(3):329–366, 2004.

[Bac01] R. Backhouse. Mathematics and programming. a revolution in the art of effective reasoning. Inaugural Lecture, School of Computer Science and IT, University of Nottingham, 2001.

[Bar80] H. Barendregt. *The Lambda-Calculus: Its Syntax and Semantics*. Elsevier Science Publishers B. V. (North-Holland), 1980.

[BCF02] N. Benton, L. Cardelli, and C. Fournet. Modern Concurrency Abstractions for C#. In Boris Magnusson, editor, *Proc. of ECOOP 2002*, pages 415–440. Springer Lect. Notes Comp. Sci. (2374), 2002.

[BK85] J. Bergstra and J. Klop. Algebra for communicating processes with abstraction. *Theoretical Computer Science*, (37):77–121, 1985.

[CG98] L. Cardelli and A. D. Gordon. Mobile ambients. In N. Nivat, editor, *Proc. First Inter. Conf. on Foundations of Software Science and Computation Structure*, pages 140–155. Springer Lect. Notes Comp. Sci. (1378), 1998.

[Cho56] N. Chomsky. Three models for the description of language. *IRE Trans. on Information Theory*, 2:113–124, 1956.

[Chu36] A. Church. An unsolvable problem of elementary number theory. *American Journal of Mathematics*, 58:345–363, 1936.

[Chu41] A. Church. *The Calculi of Lambda Conversion*. Princeton University Press, 1941.

[DMN68] O.-J. Dahl, B Myhrhang, and K. Nygaard. The SIMULA 67 Common Base Language. Tech. Report, Norvegian Computing Center, 1968.

[Fia04] J. L. Fiadeiro. Software services: scientific challenge or industrial hype? In K. Araki and Z. Liu, editors, *Proc. First International Colloquim on Theoretical Aspects of Computing (ICTAC'04), Guiyang, China*, pages 1–13. Springer Lect. Notes Comp. Sci. (3407), 2004.

[Hen88] M. C. Hennessy. *Algebraic Theory of Processes*. Series in the Foundations of Computing. MIT Press, 1988.

[HM85] M. C. Hennessy and A. J. R. G. Milner. Algebraic laws for non-determinism and concurrency. *Journal of ACM*, 32(1):137–161, 1985.

[Hoa78] C. A. R. Hoare. Communicating sequential processes. *Communications of the ACM*, 21(8):666–677, 1978.

[Hoa85] C. A. R Hoare. *Communicating Sequential Processes*. Series in Computer Science. Prentice-Hall International, 1985.

[HP85] D. Harel and A. Pnueli. On the development of reactive systems. In *Logics and Models of Concurrent Systems*, volume 13 of *NATO Adv. Sci. Inst. Ser. F Comput. Systems Sci.*, pages 477–498. Springer-Verlag, 1985.

[Kle35] S. Kleene. A theory of positive integers in formal logic. *American Journal of Mathematics*, 57:219–244, 1935.

[Kle56] S. Kleene. Representation of events in nerve nets and finite automata. In C. E. Shannon and J. McCarthy, editors, *Automata Studies*, pages 3–42. Princeton University Press, 1956.

[Koz83] D. Kozen. Results on the propositional μ-calculus. *Theoretical Computer Science*, (27):333–353, 1983.

[Mil80] R. Milner. *A Calculus of Communicating Systems*. Springer Lect. Notes Comp. Sci. (92), 1980.

[Mil89] R. Milner. *Communication and Concurrency*. Series in Computer Science. Prentice-Hall International, 1989.

[Mil97] R. Milner. Turing, computing and communication. In Recollection of Alan Turing, King's College, Cambridge University, 1997.

[Mil99] R. Milner. *Communicating and Mobile Processes: the π-Calculus*. Cambridge University Press, 1999.

[MPW92] R. Milner, J. Parrow, and D. Walker. A calculus of mobile processes (parts i and ii). *Information and Computation*, (100):1–77, 1992.

[Par01] J. Parrow. An introduction to the π-calculus. In J. Bergstra, A. Ponse, and S. Smolka, editors, *Handbook of Process Algebra*. Elsevier, 2001.

[Pet62] C. A. Petri. *Kommunikation mit Automaten*. PhD thesis, Technische Hochschule Darmstadt, 1962.

[PG03] M. P. Papazoglou and D. Georgakopoulos. Service-oriented computing. *Comms. of the ACM*, 46(10), 2003.

[Rei88] W. Reisig. Temporal logic and causality in concurrent systems. In F. H. Vogt, editor, *Concurrency 88*. LNCS (335), 1988.

[Sti92] C. Stirling. Modal and temporal logics. In Maibaum Abramsky, Gabbay, editor, *Handbook of Logic in Computer Science (vol. 2)*, pages 478–551. Oxford Science Publications, 1992.

[SW01] D. Sangiorgi and D. Walker. *The π-calculus: A Theory of Mobile Processes*. Cambridge University Press, 2001.

[Tur37] A. Turing. On computable numbers, with an application to the entscheidungsproblem. *Proc. Lond. Math. Soc.*, 42(2):230–265, 1937.

[VM94] B. Victor and F. Moller. The mobility workbench: A tool for the π-calculus. In *Proceedings of CAV'94: Computer Aided Verification*. Springer Lect. Notes Comp. Sci. (818), 1994.

[WN95] G. Winskel and M. Nielsen. Models for Concurrency. In S. Abramsky, D. M. Gabbay, and T. S. E. Gabbay, editors, *Handbook of Logic in Computer Science (vol. 4))*, pages 1–148, 1995.