Guião para aula laboratorial de Verificação Formal (2019/20)

Coq (1)

O objectivo desta aula é a familiarização com o sistema Coq, um sistema de gestão e desenvolvimento de prova formal.

Deverá ter instalado o CoqIde. Como alternativa, se for um utilizador do Emacs, poderá instalar o Coq e o modo `Proof-General` para o Emacs (e opcionalmente o `Company-Coq`).

O website do Coq disponibiliza toda a documentação, assim como diversos livros e tutoriais que fornecem uma introdução rápida ao sistema Coq.

1 Aspectos básicos do desenvolvimento de provas em Coq

Vamos ilustrar os aspectos mais básicos do Coq como ferramenta de prova, tendo por base exemplos da lógica proposicional e da lógica de 1ª ordem que se demonstram com base num raciocínio deductivo.

Comece por invocar o CoqIde e carregue o ficheiro `lesson1.v`.

```
$ coqide &
```

Neste ficheiro pretendemos ir apresentado a sintaxe do Coq e ilustrar o funcionamento de vários comandos e tácticas de prova através de vários exemplos comentados.

Execute, passo a passo, as instruções deste ficheiro e analize o seu efeito. Atente nos comentários lá colocados e no efeito da aplicação de cada táctica de prova na evolução do estado da prova.

Recorde algumas das **tácticas básicas** de prova:

Proposition (P)	**Introduction**	**Elimination** (H of type P)
$\bot$		`elim` H, `contradiction`
$\neg A$	`intro`	`apply` H
$A \wedge B$	`split`	`elim` H, `destruct` H `as [H1 H2]`
$A \Rightarrow B$	`intro`	`apply` H
$A \vee B$	`left`, `right`	`elim` H, `destruct` H `as [H1|H2]`
$\forall x{:}A.\, Q$	`intro`	`apply` H
$\exists x{:}A.\, Q$	`exists` *witness*	`elim` H, `destruct` H `as [x H1]`

`intro`, `intros` – regra de introdução para Π (várias vezes).

`apply` – regra de eliminação para Π.

`assumption` – combinar a conclusão com uma hipótese.

`exact` – apresenta diretamente um termo de prova exacto para a conclusão.

Recorde algumas **tácticas automáticas** de prova:

`trivial` – tenta as tácticas que podem resolver aprova num passo.

`auto` – tenta uma combinação de tácticas `intro`, `apply` e `assumption` usando os teoremas armazenados numa base de dados como dicas para essa táctica.

`tauto`, `intuition` – útil para provar tautologias da lógica proposicional intuicionista.

`firstorder` – útil para provar fatos que são tautologias da lógica de 1ª ordem intuicionista.

Complete as provas em falta, substituindo o comando `Admitted` e por uma script de prova apropriada.

Crie agora um novo ficheiro Coq para desenvolver as provas das propriedades que se seguem.

2 Lógica proposicional

Prove as seguintes tautologias da lógica proposicional:

1. $(A \vee B) \vee C \to A \vee (B \vee C)$
2. $(B \to C) \to A \vee B \to A \vee C$
3. $(A \wedge B) \wedge C \to A \wedge (B \wedge C)$
4. $A \vee (B \wedge C) \to (A \vee B) \wedge (A \vee C)$
5. $(A \wedge B) \vee (A \wedge C) \leftrightarrow A \wedge (B \vee C)$
6. $(A \vee B) \wedge (A \vee C) \leftrightarrow A \vee (B \wedge C)$

3 Lógica de primeira ordem

Prove os seguintes teoremas da lógica de primeira ordem:

1. $(\exists x.P(x) \wedge Q(x)) \to (\exists x.P(x)) \wedge (\exists x.Q(x))$
2. $(\exists x.\forall y.P(x,y)) \to \forall y.\exists x.P(x,y)$
3. $(\exists x.P(x)) \to (\forall x.\forall y.P(x) \to Q(y)) \to \forall y.Q(y)$
4. $(\forall x.Q(x) \to R(x)) \to (\exists x.P(x) \wedge Q(x)) \to \exists x.P(x) \wedge R(x)$
5. $(\forall x.P(x) \to Q(x)) \to (\exists x.P(x)) \to \exists y.Q(y)$
6. $(\exists x.P(x)) \vee (\exists x.Q(x)) \leftrightarrow (\exists x.P(x) \vee Q(x))$

4 Lógica clássica

Assumindo o *princípio do meio excluído* como axioma, prove que:

1. $((A \to B) \to A) \to A \quad (\textit{lema de Pierce})$
2. $\neg\neg A \to A$
3. $\neg\forall x.P(x) \to \exists x.\neg P(x)$