Testing and Verification

We wish to establish that a program is correct.
That is to say, it meets its specifications.

Testing

> Try a selection of inputs and check
> against results

There is no guarantee that all bugs will be found.
Behaviour of a program is not continuous

Verification

> Prove that the program is correct.

Proofs can be long, tedious and complicated.
Functional programs are easier to do proofs with.

Formal vs. Rigorous proof

Formal Proof:

- what logicians study
- purely symbolic
- needs an axiom system
- needs machine support

Rigorous Proof:

- what mathematicians do
- in natural language (with some symbols)
- needs clear foundations
- understandable to people

A rigorous proof is a convincing mathematical argument.

Assumptions

In order to reason with ML programs, we make some assumptions about the expressions that will be used in proofs.

- Expressions are purely functional.
- Expressions are well-formed and well-typed.
- Types are interpreted as sets.
- Execution of programs always terminates.

```
fun undef(x) = undef(x + 1);
```

`factorial` terminates for $n \geq 0$.

Principle of Induction

We have a property $\phi(n)$ we wish to prove for all integers $n \geq 0$.

Prove:

- $\phi(0)$; and
- $\phi(k)$ implies $\phi(k+1)$, for all k.

This proves $\phi(n)$ for all n.

For any particular n, $\phi(n)$ can be derived in n steps.

Two Factorial Functions

```
fun fact n =
    if n = 0 then 1 else n*(fact n-1);
```

Or, a more efficient, iterative, version

```
fun facti (n,p) =
    if n=0 then p else
            facti(n-1, n*p);
```

Are the two functions equivalent?

Can we show that for all n:

$$\texttt{facti}(n, 1) = \texttt{fact}(n)$$

Inductive Proof

We make the stronger inductive hypothesis.

$$\forall p\ \texttt{facti}(n, p) = \texttt{fact}(n) \times p$$

Base case:

$$\begin{aligned}
\texttt{facti}(0, p) &= p && \text{defn. of } \texttt{facti} \\
&= 1 \times p \\
&= \texttt{fact}(0) \times p && \text{defn. of } \texttt{fact}
\end{aligned}$$

Induction Step:

$$\begin{aligned}
\texttt{facti}(k+1, p) &= \texttt{facti}(k, (k+1) \times p) \\
&= \texttt{fact}(k) \times (k+1) \times p \\
&= \texttt{fact}(k+1) \times p
\end{aligned}$$

Complete Induction

To show that $\phi(n)$ is true for all n, it is also sufficient to show that:

$$(\forall i < k\, \phi(i)) \text{ implies } \phi(k)$$

- no separate base case is necessary
- $\phi(n)$ is still provable in n steps
- a more general form of proof by induction

Proof by Complete Induction

```
fun power(x,k) =
    if k=1 then x
    else if k mod 2 = 0 then
         power(x*x, k div 2)
    else x*power(x*x, k div 2);
```

Prove by induction (on k) that for all $k \geq 1$:

$$\forall x \; \texttt{power}(x, k) = x^k$$

Complete Induction (*contd.*)

if $k = 1$ then $\texttt{power}(x, 1) = x = x^1$

if $k = 2j$ then

$$\begin{aligned} \texttt{power}(x, k) &= \texttt{power}(x, 2j) \\ &= \texttt{power}(x^2, j) \\ &= (x^2)^j \\ &= x^{2j} \end{aligned}$$

if $k = 2j + 1$ then

$$\begin{aligned} \texttt{power}(x, k) &= \texttt{power}(x, 2j + 1) \\ &= x \times \texttt{power}(x^2, j) \\ &= x \times (x^2)^j \\ &= x^{2j+1} \end{aligned}$$