Lecture # 12

Temporal Logic

Intuition

In classical propositional logic , formulae are evaluated within a single fixed world (or state). For example, a proposition such as

"it is Monday"

is either *true* or *false*. Such propositions are then combined using constructs (or connectives) such as '$\wedge$', '$\neg$', and '$\Rightarrow$' Classical propositional logic is thus used to describe *static* situations. The meaning of statements in such a logic can then be modelled by mapping basic (also called *atomic*) propositions to Boolean values (i.e., **T** or **F**). For example,

$$[a \mapsto \mathbf{T},\ b \mapsto \mathbf{F},\ c \mapsto \mathbf{T}]$$

is a particular structure (called a 'model') satisfying the propositional formula

$$a \wedge (b \vee c).$$

Intuition

Here, the model states that b is **F**alse, yet c is **T**rue. This means that $b \vee c$ is **T**rue and, combined with the fact that a is **T**rue, means that, overall, $a \wedge (b \vee c)$ is **T**rue for this allocation of truth values to atomic propositions. There are, of course, other models that satisfy this formula, for example

$$[\, a \mapsto \mathbf{T},\ b \mapsto \mathbf{T},\ c \mapsto \mathbf{F} \,].$$

For any such model, M, we can write that

$$M \models a \wedge (b \vee c)$$

meaning that 'the model M satisfies the formula $a \wedge (b \vee c)$'.

Temporal Logic

- In classical logic, formulae are evaluated within a single fixed world. For example, a proposition such as “it is Monday” is either true or false.
- Propositions are then combined using constructs such as ‘$\wedge$’, ‘$\neg$’, etc.
- In temporal logics, evaluation takes place within a set of worlds. Thus, “it is Monday” may be satisfied in some worlds, but not in others.

Temporal Logic

The idea of temporal logic is that a formula is not statically true or false in a model, as it is in propositional and predicate logic.

Instead, the models of temporal logic contain several states and a formula can be true in some states and false in others.

Thus, the static notion of truth is replaced by a dynamic one, in which the formulas may change their truth values as the system evolves from state to state.

Kripke Model

As we have seen, temporal logic is an extension of classical logic, whereby time becomes an extra parameter modifying the truth of logical statements. Models for temporal (and modal) logics are typically 'Kripke Structures' [325] of the form

$$\mathcal{M} = \langle S, R, \pi \rangle$$

where

- S is the set of *moments* in time (our accessible worlds or states),
- R is a temporal accessibility relation (in the case of PTL, this characterizes a sequence that is linear, discrete and has finite past), and
- $\pi : S \mapsto \mathbf{P}(\text{PROP})$ maps each moment/world/state to a set of propositions[3] (i.e. those that are true in that moment/world/state).

However, in the case of PTL, which has a linear, discrete basis that is isomorphic to $\mathbb{N}$, this model structure is often simplified from the above to

$$\mathcal{M} = \langle \mathbb{N}, \pi \rangle$$

where

- $\pi : \mathbb{N} \mapsto \mathbf{P}(\text{PROP})$ maps each Natural Number (representing a moment in time) to the set of propositions true at that moment.

[3] '**P**' is the *powerset* operator, which takes a set, say X, and generates a new set containing all possible subsets of X. For example, $\mathbf{P}(\{a, b, c\}) = \{\emptyset, \{a\}, \{b\}, \{c\}, \{a, b\}, \{a, c\}, \{b, c\}, \{a, b, c\}\}$.

Kripke Model

Definition : A model $\mathcal{M}$ of basic modal logic is specified by three things:

1. A set W, whose elements are called worlds;
2. A relation R on W ($R \subseteq W \times W$), called the accessibility relation;
3. A function $L : W \to \mathcal{P}(\texttt{Atoms})$, called the labelling function.

We write $R(x, y)$ to denote that (x, y) is in R.

These models are often called Kripke models

Kripke Model

Suppose **W** equals **{x1,x2,x3,x4,x5,x6}** and the relation **R** is given as follows:

R(x1,x2), R(x1,x3), R(x2,x2), R(x2,x3), R(x3,x2), R(x4,x5), R(x5,x4), R(x5,x6); and no other pairs are related by R.

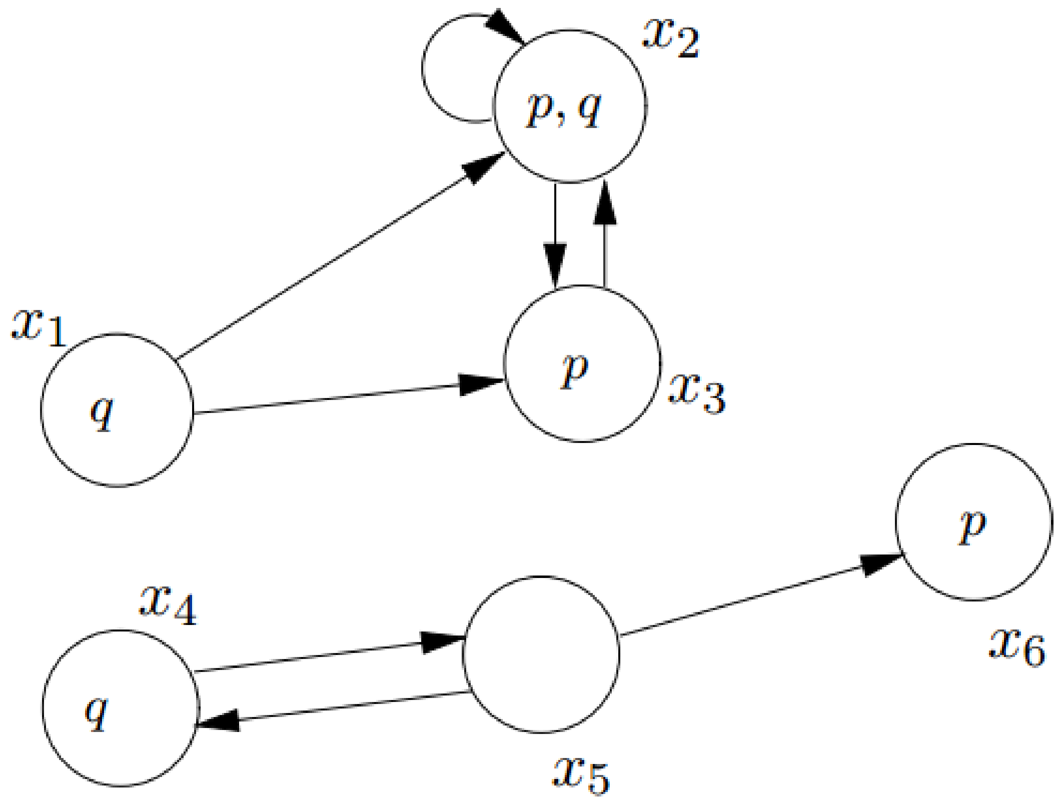
Suppose further that the labelling function behaves as follows:

x	x_1	x_2	x_3	x_4	x_5	x_6
$L(x)$	$\{q\}$	$\{p, q\}$	$\{p\}$	$\{q\}$	$\emptyset$	$\{p\}$

Kripke Model

The set **W** is drawn as a set of circles, with arrows between them showing the relation **R**.

Within each circle is the value of the labelling function in that world.

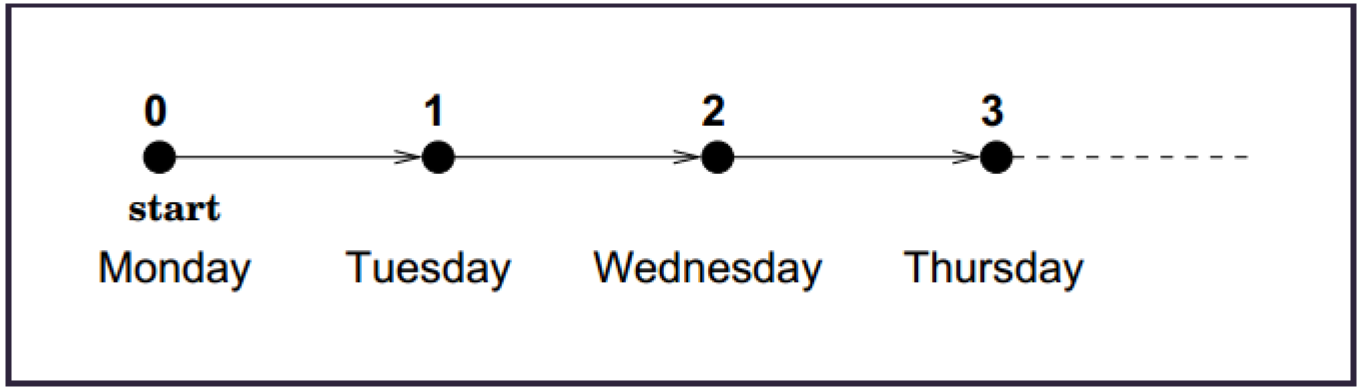


PTL

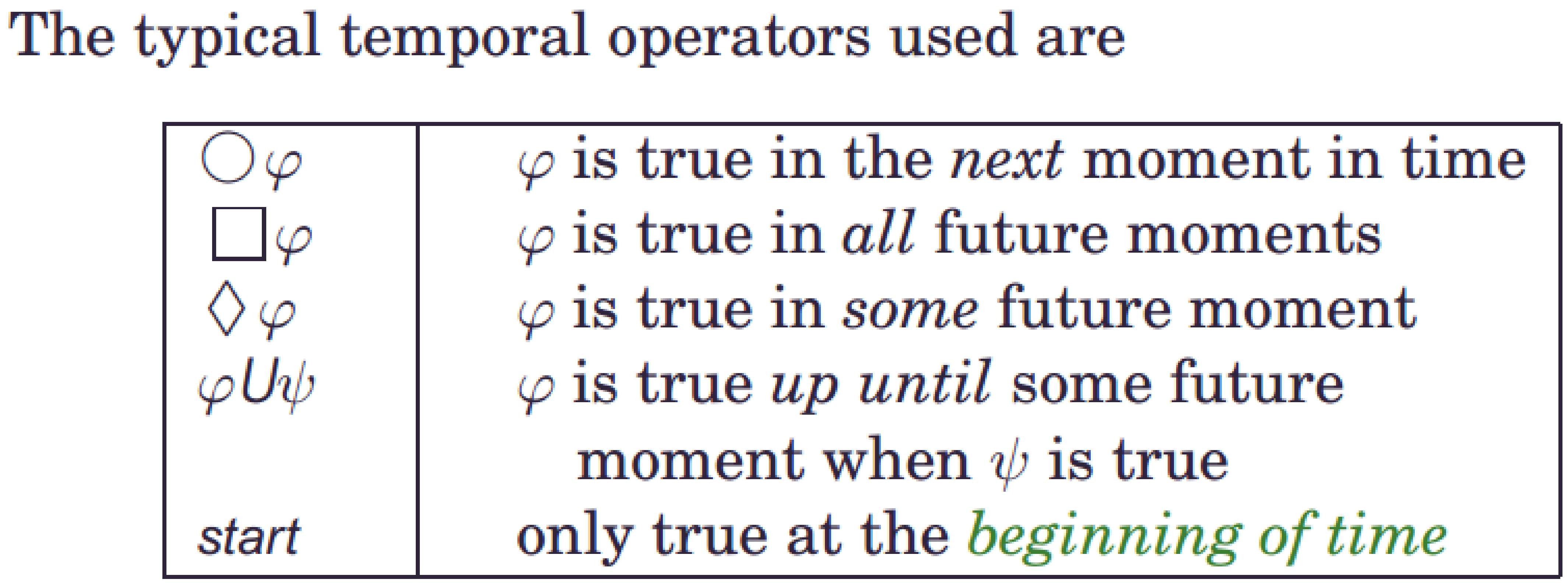
PTL (Propositional Temporal Logic) also called as Liner Temporal Logic LTL.

We use a simple temporal logic (PTL) where the accessibility relation characterizes a discrete, linear order isomorphic to the Natural Numbers, N

If we consider days within the week, beginning with Monday, we might get the discrete sequence



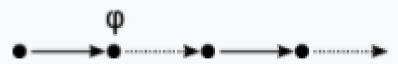
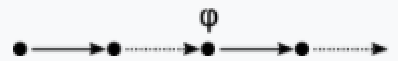
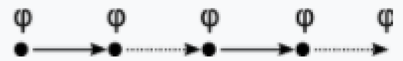
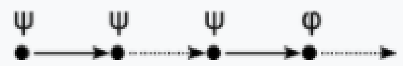
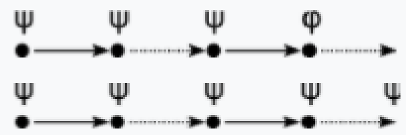
Temporal Logic



The typical temporal operators used are

$\bigcirc\varphi$	φ is true in the *next* moment in time
$\square\varphi$	φ is true in *all* future moments
$\lozenge\varphi$	φ is true in *some* future moment
$\varphi U \psi$	φ is true *up until* some future moment when ψ is true
start	only true at the *beginning of time*

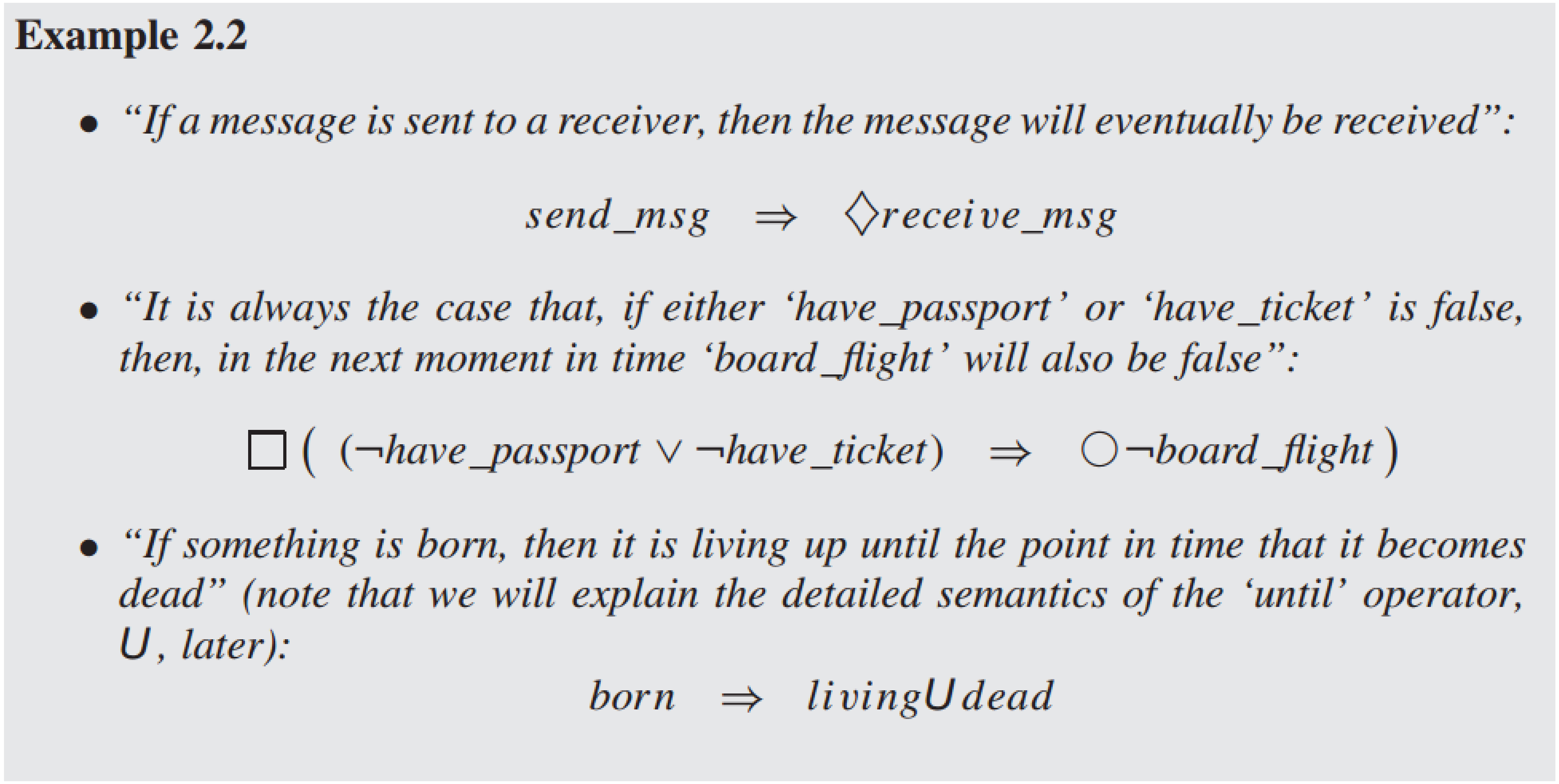
Temporal Logic

Textual	Symbolic	Explanation	Diagram
Unary operators:			
X φ	$\bigcirc\varphi$	ne**X**t: φ has to hold at the next state.	
F φ	$\Diamond\varphi$	**F**inally: φ eventually has to hold (somewhere on the subsequent path).	
G φ	$\Box\varphi$	**G**lobally: φ has to hold on the entire subsequent path.	
Binary operators:			
ψ **U** φ	$\psi\,\mathcal{U}\,\varphi$	**U**ntil: ψ has to hold *at least* until φ becomes true, which must hold at the current or a future position.	
ψ **W** φ	$\psi\,\mathcal{W}\,\varphi$	**W**eak until: ψ has to hold *at least* until φ; if φ never becomes true, ψ must remain true forever.	

PTL Examples

$$\begin{aligned}
\mathit{monday} &\Rightarrow \bigcirc\, \mathit{tuesday} \\
\mathit{start} &\Rightarrow \Diamond\, \mathit{finish} \\
\mathit{july} &\Rightarrow \Diamond\, (\mathit{december} \wedge \mathit{winter}) \\
\mathit{send}(\mathit{msg}, \mathit{rcvr}) &\Rightarrow \Diamond\, \mathit{receive}(\mathit{msg}, \mathit{rcvr}) \\
\Box((\neg \mathit{passport} \vee \neg \mathit{ticket}) &\Rightarrow \bigcirc\, \neg \mathit{board\_flight}) \\
\mathit{sunset} &\Rightarrow \bigcirc\, (\mathit{night}\, \mathcal{U}\, \mathit{dawn}) \\
\mathit{born} &\Rightarrow \Diamond\, \Box\, \mathit{old} \\
\mathit{monday} &\Rightarrow \mathit{sad}\, \mathcal{U}\, \mathit{saturday}
\end{aligned}$$

PTL EXAMPLE



Example 2.2

- *"If a message is sent to a receiver, then the message will eventually be received":*

$$send\_msg \quad \Rightarrow \quad \Diamond receive\_msg$$

- *"It is always the case that, if either 'have_passport' or 'have_ticket' is false, then, in the next moment in time 'board_flight' will also be false":*

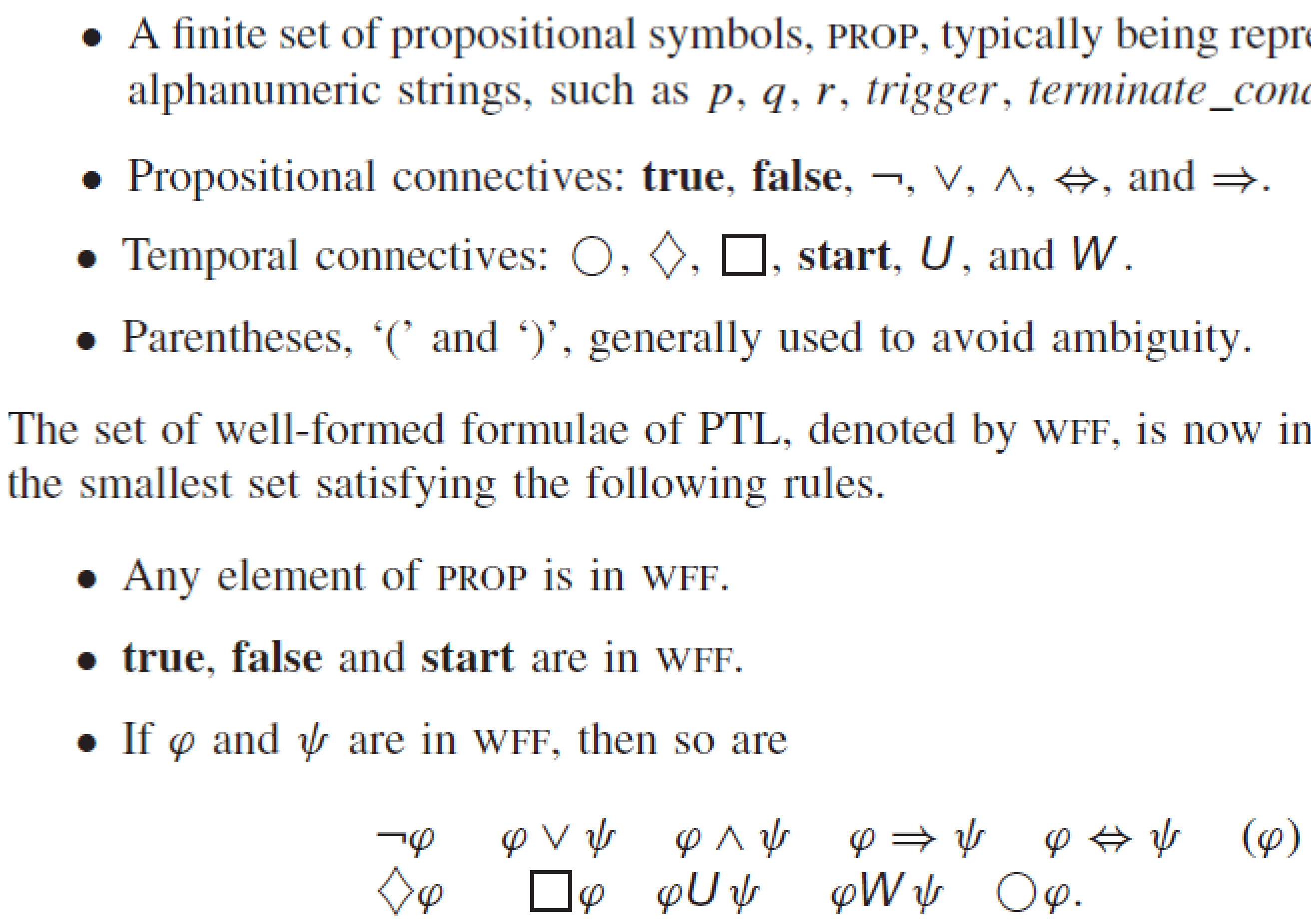
$$\Box\big(\ (\neg have\_passport \lor \neg have\_ticket) \quad \Rightarrow \quad \bigcirc \neg board\_flight\ \big)$$

- *"If something is born, then it is living up until the point in time that it becomes dead" (note that we will explain the detailed semantics of the 'until' operator, $\mathcal{U}$, later):*

$$born \quad \Rightarrow \quad living\,\mathcal{U}\,dead$$

Formal definition

We now describe the formal syntax of PTL; Formulae in PTL are constructed from the following elements.



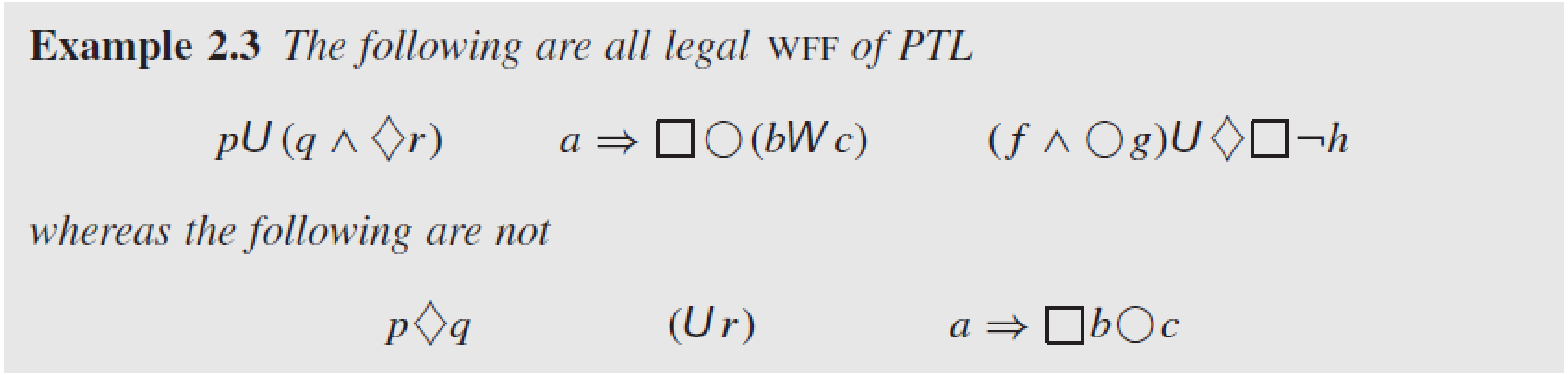
- A finite set of propositional symbols, PROP, typically being represented by lowercase alphanumeric strings, such as p, q, r, *trigger*, *terminate_condition2*, *lunch*, ...
- Propositional connectives: **true**, **false**, $\neg$, $\vee$, $\wedge$, $\Leftrightarrow$, and $\Rightarrow$.
- Temporal connectives: $\bigcirc$, $\Diamond$, $\Box$, **start**, $\mathcal{U}$, and $\mathcal{W}$.
- Parentheses, '(' and ')', generally used to avoid ambiguity.

The set of well-formed formulae of PTL, denoted by WFF, is now inductively defined as the smallest set satisfying the following rules.

- Any element of PROP is in WFF.
- **true**, **false** and **start** are in WFF.
- If φ and ψ are in WFF, then so are

$$\neg\varphi \quad \varphi \vee \psi \quad \varphi \wedge \psi \quad \varphi \Rightarrow \psi \quad \varphi \Leftrightarrow \psi \quad (\varphi)$$
$$\Diamond\varphi \quad \Box\varphi \quad \varphi\mathcal{U}\psi \quad \varphi\mathcal{W}\psi \quad \bigcirc\varphi.$$

Formal definition



Example 2.3 *The following are all legal* WFF *of PTL*

$$p\,\mathcal{U}\,(q \wedge \Diamond r) \qquad a \Rightarrow \Box \bigcirc (b\,\mathcal{W}\,c) \qquad (f \wedge \bigcirc g)\,\mathcal{U}\,\Diamond \Box \neg h$$

whereas the following are not

$$p \Diamond q \qquad (\mathcal{U}\, r) \qquad a \Rightarrow \Box b \bigcirc c$$

Working with PTL

Model Checking

Very popular technique for checking whether a temporal formula is satisfied in a particular (finite) structure

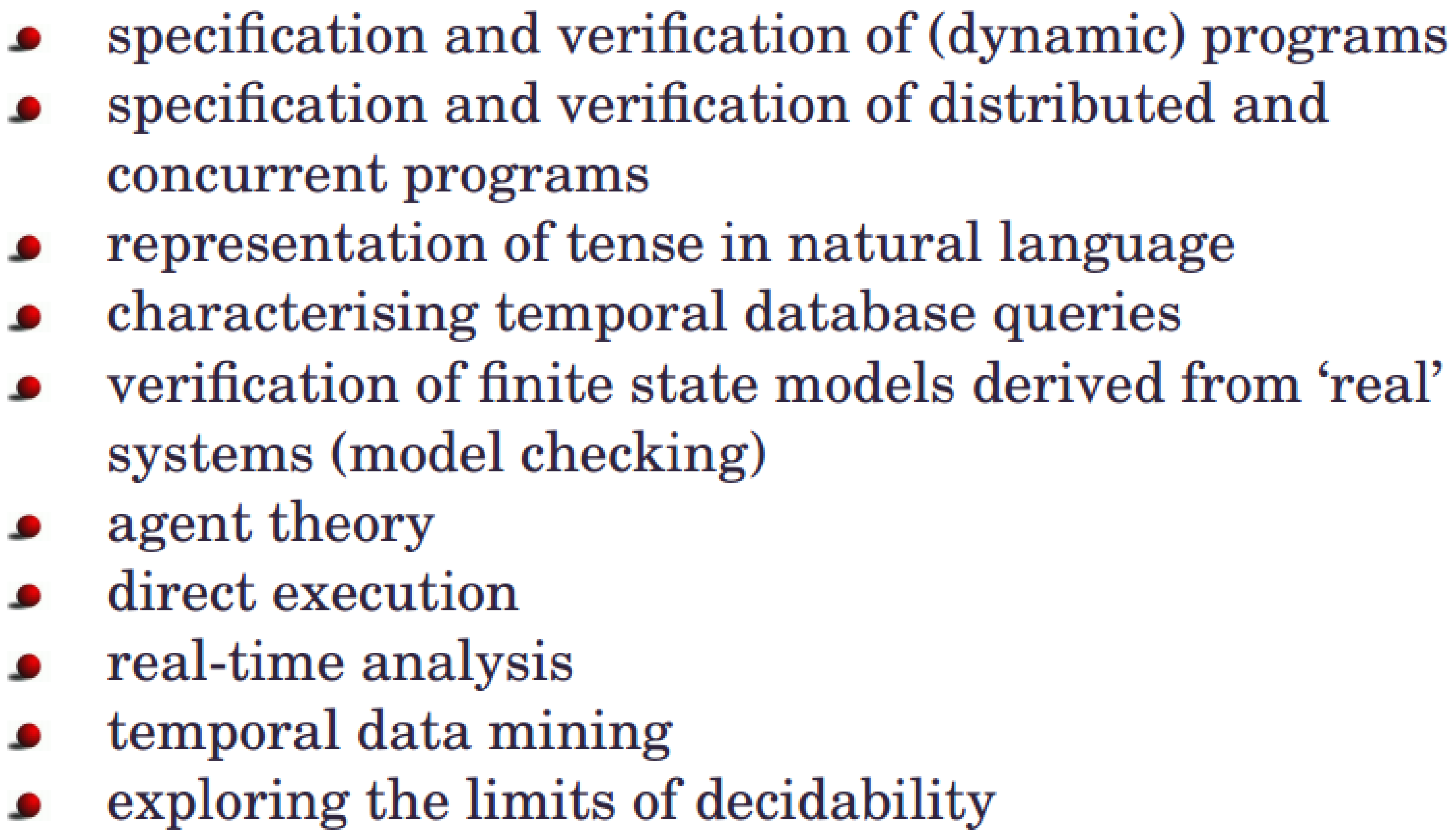
- the finite structure is often derived from program or hardware descriptions.

Automata

Temporal logics have a close relationship to ω automata, i.e. finite state automata over infinite objects

- consequently automata-theoretic methods are often employed.

Temporal Logic Gets Everywhere



- specification and verification of (dynamic) programs
- specification and verification of distributed and concurrent programs
- representation of tense in natural language
- characterising temporal database queries
- verification of finite state models derived from ‘real’ systems (model checking)
- agent theory
- direct execution
- real-time analysis
- temporal data mining
- exploring the limits of decidability

Intuition: Safety Properties

Safety:

"something bad will *not* happen"

Typical examples:

$\square \neg(\mathit{reactor\_temp} > 1000)$

$\square \neg(\mathit{one\_way} \wedge \bigcirc \mathit{other\_way})$

$\square \neg((x = 0) \wedge \bigcirc\bigcirc\bigcirc(y = z/x))$

and so on.....

Usually: $\square \neg \ldots$

Intuition: Liveness Properties

Liveness:

“something good *will* happen”

Typical examples:

$\Diamond$ *rich*

$\Diamond$ *terminate*

$\Diamond (x > 5)$

and so on.....

Usually: $\Diamond \ldots$.

Intuition: Fairness Properties

Fairness (strong):

> "if we attempt/request infinitely often, then we will be successful/allocated infinitely often"

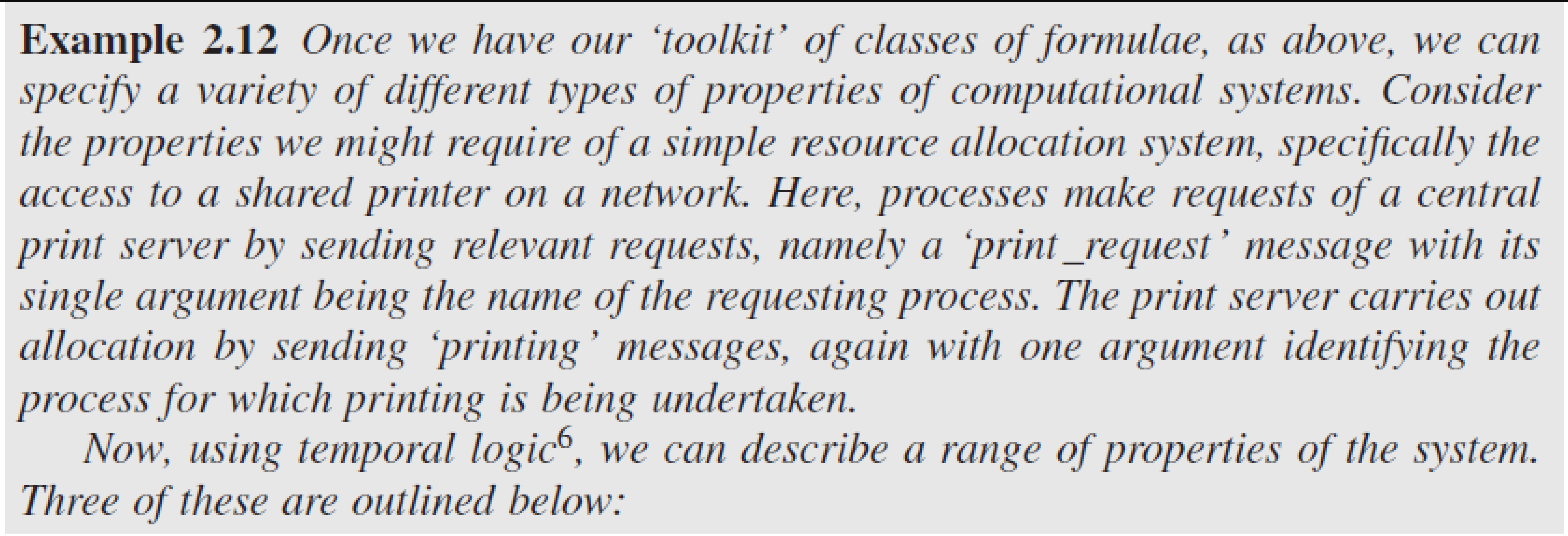
Often only really useful when scheduling processes, responding to messages, etc. Typical example:

$$\forall p \in \mathit{processes}.\ \Box \Diamond \mathit{ready}(p) \Rightarrow \Box \Diamond \mathit{run}(p)$$

There are many forms of fairness, e.g:

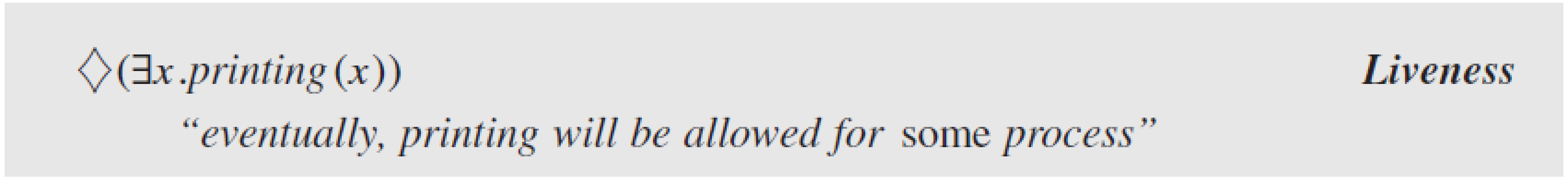
$$\Box \Diamond \mathit{attempt} \Rightarrow \Box \Diamond \mathit{succeed}$$
$$\Box \Diamond \mathit{attempt} \Rightarrow \Diamond \mathit{succeed}$$
$$\Box \mathit{attempt} \Rightarrow \Box \Diamond \mathit{succeed}$$
$$\Box \mathit{attempt} \Rightarrow \Diamond \mathit{succeed}$$

Reactive system properties



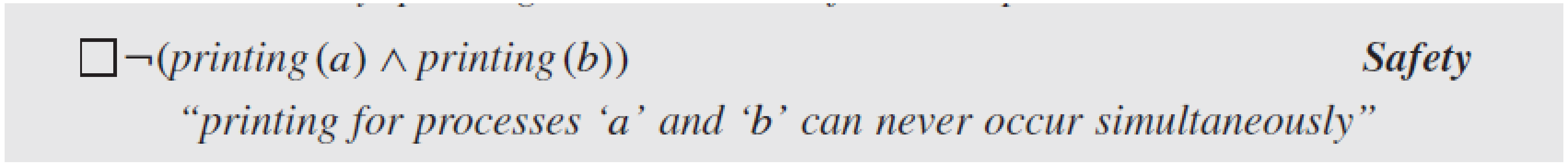
Example 2.12 *Once we have our 'toolkit' of classes of formulae, as above, we can specify a variety of different types of properties of computational systems. Consider the properties we might require of a simple resource allocation system, specifically the access to a shared printer on a network. Here, processes make requests of a central print server by sending relevant requests, namely a 'print_request' message with its single argument being the name of the requesting process. The print server carries out allocation by sending 'printing' messages, again with one argument identifying the process for which printing is being undertaken.*

Now, using temporal logic[6]*, we can describe a range of properties of the system. Three of these are outlined below:*



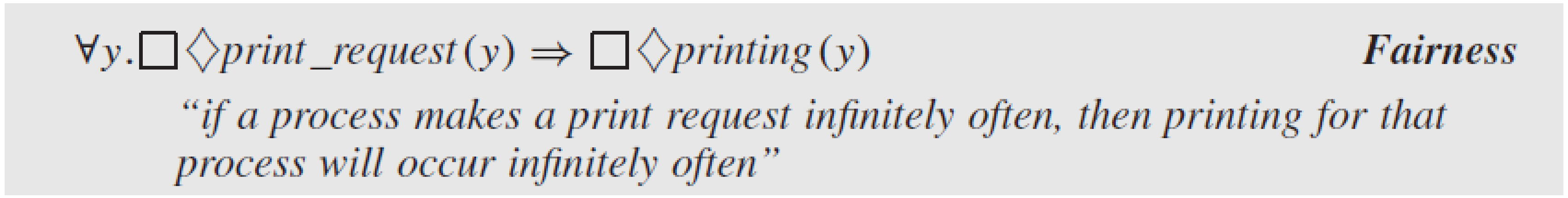
$\Diamond(\exists x.\, printing(x))$ **_Liveness_**

"eventually, printing will be allowed for some *process"*



$\Box\neg(printing(a) \wedge printing(b))$ **_Safety_**

"printing for processes 'a' and 'b' can never occur simultaneously"



$\forall y.\Box\Diamond print\_request(y) \Rightarrow \Box\Diamond printing(y)$ **_Fairness_**

"if a process makes a print request infinitely often, then printing for that process will occur infinitely often"

REFERENCES :

1. Michael Huth, Mark Ryan-Cambridge, Logic in computer science_ modelling and reasoning about systems University Press (2004) - Chapter 5