Utilizando Lógicas Deónticas y Temporales para la Especificación de Sistemas Tolerantes a Fallas

Departamento de Computación
Facultad de Ciencias Exactas, Físico-Químicas y Naturales
Universidad Nacional de Río Cuarto
ckilmurray AT dc.exa.unrc.edu.ar

I Jornada de Doctorandos en Ciencias de la Computación de FaMAF

Cecilia Kilmurray

Cecilia Kilmurray

- Licenciada en Ciencias de la Computación (2005).
- Motorola (2006-2009).
- Docente del Departamento de computación de la UNRC desde Abril de 2010.
- Estudiante de Doctorado desde Junio de 2010. Bajo la dirección de *Pablo Castro* en la UNRC.
- Trabajo en el area de Sistemas Tolerantes a Fallas, en particular la idea es extender alguna de las Lógicas Deónticas, incluyendo alguna noción de Probabilidad de manera tal que nos permita especificar y analizar propiedades de este tipo de sistemas.

Qué es un Sistema Tolerante a Fallos?

Es la capacidad de un sistema de proveer cierto comportamiento esperado aún ante la ocurrencia ocasional de fallas.

Qué es un Sistema Tolerante a Fallos?

Es la capacidad de un sistema de proveer cierto comportamiento esperado aún ante la ocurrencia ocasional de fallas.

Qué es un Sistema Tolerante a Fallos?

Es la capacidad de un sistema de proveer cierto comportamiento esperado aún ante la ocurrencia ocasional de fallas.

Qué es un Sistema Tolerante a Fallos?

Es la capacidad de un sistema de proveer cierto comportamiento esperado aún ante la ocurrencia ocasional de fallas.

Ejemplo: Generales Bizantinos

Ejemplo: Generales Bizantinos

Generales Bizantinos

- Varias divisiones del ejército Bizantino alrededor de una ciudad.
- Cada división tiene un **General** que esta a cargo de la misma.
- Deben decidir entre todos un plan de acción en común.
- Hay generales que son **Traidores**.

Ejemplo: Generales Bizantinos

Generales Bizantinos

- Varias divisiones del ejército Bizantino alrededor de una ciudad.
- Cada división tiene un **General** que esta a cargo de la misma.
- Deben decidir entre todos un plan de acción en común.
- Hay generales que son **Traidores**.

Los generales deben encontrar un algoritmo que garantice:

Ejemplo: Generales Bizantinos

Generales Bizantinos

- Varias divisiones del ejército Bizantino alrededor de una ciudad.
- Cada división tiene un **General** que esta a cargo de la misma.
- Deben decidir entre todos un plan de acción en común.
- Hay generales que son **Traidores**.

Los generales deben encontrar un algoritmo que garantice:

- Que todos los generales **Leales** decidan el mismo plan de acción.

Ejemplo: Generales Bizantinos

Generales Bizantinos

- Varias divisiones del ejército Bizantino alrededor de una ciudad.
- Cada división tiene un **General** que esta a cargo de la misma.
- Deben decidir entre todos un plan de acción en común.
- Hay generales que son **Traidores**.

Los generales deben encontrar un algoritmo que garantice:

- Que todos los generales **Leales** decidan el mismo plan de acción.
- El pequeño grupo de **Traidores** no pueda causar que el resto de los generales decidan un plan incorrecto.

Ejemplo: Generales Bizantinos (cont.)

El Problema de los Generales Bizantinos

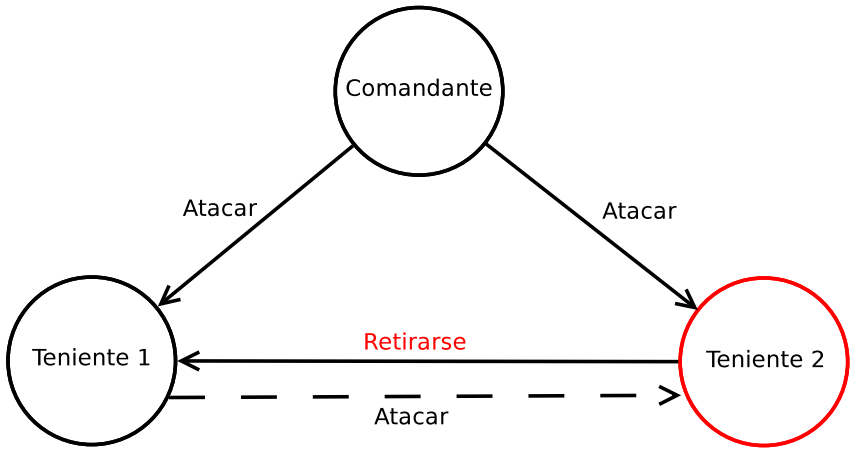
- Un Comandante General debe enviar una orden a sus **n-1** tenientes.
  - Todos los tenientes **Leales** obedecen la misma orden.
  - Si el Comandante es **Leal**, cada teniente leal obedece la orden que el envió.

Ejemplo: Generales Bizantinos (cont.)

El Problema de los Generales Bizantinos

- Un Comandante General debe enviar una orden a sus **n-1** tenientes.
  - Todos los tenientes **Leales** obedecen la misma orden.
  - Si el Comandante es **Leal**, cada teniente leal obedece la orden que el envió.

Consideremos **n**=3, 1 es traidor.

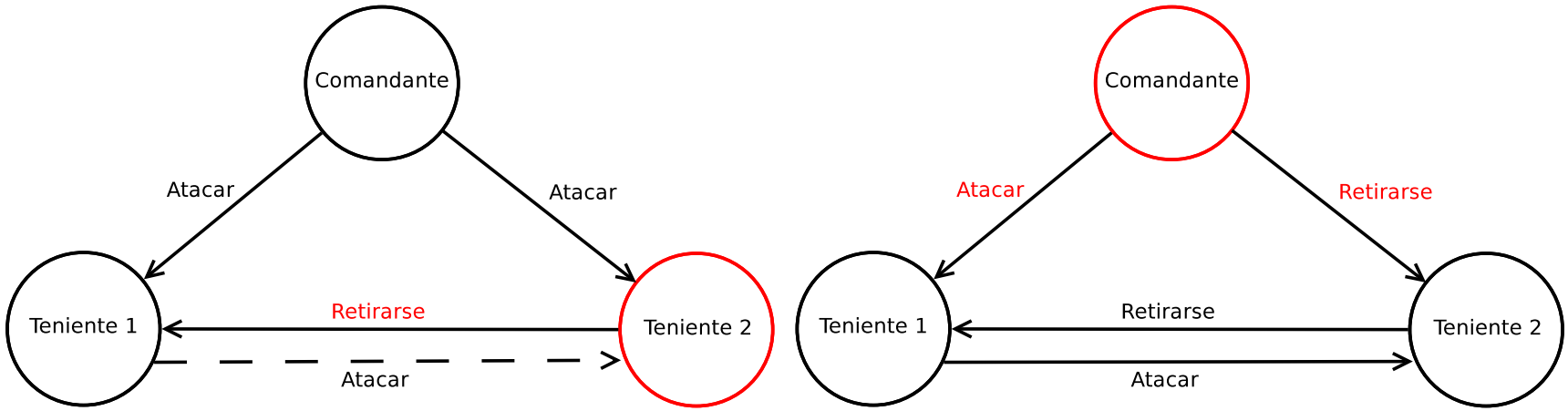


Ejemplo: Generales Bizantinos (cont.)

El Problema de los Generales Bizantinos

- Un Comandante General debe enviar una orden a sus **n-1** tenientes.
  - Todos los tenientes **Leales** obedecen la misma orden.
  - Si el Comandante es **Leal**, cada teniente leal obedece la orden que el envió.

Consideremos **n**=3, 1 es traidor.



Ejemplo: Generales Bizantinos (cont.)

No existen soluciones para 3 generales donde 1 sea *Traidor*.

Existe Solución sólo si:

- Nro. de **Traidores** $<$ **1/3** del total de Generales.
- $3m + 1$ generales pueden llegar a un acuerdo aún en la presencia de **m** traidores.

Ejemplo: Generales Bizantinos (cont.)

No existen soluciones para 3 generales donde 1 sea *Traidor*.

Existe Solución sólo si:

- Nro. de **Traidores** $<$ **1/3** del total de Generales.
- $3m + 1$ generales pueden llegar a un acuerdo aún en la presencia de **m** traidores.

Este problema abstracto muestra que cualquier problema similar puede ser reducido al siguiente conjunto de reglas:
Si se quiere tolerar hasta **m** componentes con **fallas**, vamos a necesitar:

- $3m + 1$ componentes.
- $2m + 1$ canales de comunicación independientes.
- $m + 1$ rounds de comunicación.

Lógicas Deónticas

La Lógica Deóntica es una rama de la Lógica dedicada a estudiar nociones como **permisión**, **prohibición** y **obligación**. Muchos autores utilizan los operadores modales para formalizar dichas nociones, pero con otras interpretaciones:

Lógicas Deónticas

La Lógica Deóntica es una rama de la Lógica dedicada a estudiar nociones como **permisión**, **prohibición** y **obligación**. Muchos autores utilizan los operadores modales para formalizar dichas nociones, pero con otras interpretaciones:

- $\Diamond$ significa "está permitido".

Lógicas Deónticas

La Lógica Deóntica es una rama de la Lógica dedicada a estudiar nociones como **permisión**, **prohibición** y **obligación**. Muchos autores utilizan los operadores modales para formalizar dichas nociones, pero con otras interpretaciones:

- $\Diamond$ significa "está permitido".
- $\Box$ significa "obligatoriamente"

Lógicas Deónticas

La Lógica Deóntica es una rama de la Lógica dedicada a estudiar nociones como **permisión**, **prohibición** y **obligación**. Muchos autores utilizan los operadores modales para formalizar dichas nociones, pero con otras interpretaciones:

- $\Diamond$ significa "está permitido".
- $\Box$ significa "obligatoriamente"

Una ventaja de la lógica deóntica con respecto a otros formalismos, es que nos permite distinguir el comportamiento **"correcto"** o **"normal"** del que no lo es.

Lógicas Deónticas

La Lógica Deóntica es una rama de la Lógica dedicada a estudiar nociones como **permisión**, **prohibición** y **obligación**. Muchos autores utilizan los operadores modales para formalizar dichas nociones, pero con otras interpretaciones:

- $\Diamond$ significa "está permitido".
- $\Box$ significa "obligatoriamente"

Una ventaja de la lógica deóntica con respecto a otros formalismos, es que nos permite distinguir el comportamiento **"correcto"** o **"normal"** del que no lo es.

Normative Behavior o Comportamiento Correcto

Ejecuta sólo acciones permitidas y cumple con todas sus obligaciones.

Una extensión...Lógica Deóntica con acciones

Kent, **Maibaum** y **Quirk** definen una extensión de la lógica deóntica con lógica dinámica.

Una extensión...Lógica Deóntica con acciones

Kent, **Maibaum** y **Quirk** definen una extensión de la lógica deóntica con lógica dinámica.

Especificación

Una especificación en esta lógica es vista como una comunidad de **Componentes** ó **Presentaciones de Teorías** que son combinadas a través de morfismos.

Una extensión...Lógica Deóntica con acciones

Kent, **Maibaum** y **Quirk** definen una extensión de la lógica deóntica con lógica dinámica.

Especificación

Una especificación en esta lógica es vista como una comunidad de **Componentes** ó **Presentaciones de Teorías** que son combinadas a través de morfismos.

Un **Componente** consta de 2 partes:

- **Signature**: *Atributos* + *Acciones* + *Tipos de Datos*

Una extensión...Lógica Deóntica con acciones

Kent, **Maibaum** y **Quirk** definen una extensión de la lógica deóntica con lógica dinámica.

Especificación

Una especificación en esta lógica es vista como una comunidad de **Componentes** ó **Presentaciones de Teorías** que son combinadas a través de morfismos.

Un **Componente** consta de 2 partes:

- **Signature**: *Atributos* + *Acciones* + *Tipos de Datos*
- **Theory**: Conjunto de *Fórmulas*(*axiomas*)

Fórmulas

Una fórmula puede tener la siguiente forma, donde t_1 y t_2 son **Términos**, **at** son **Términos de Acción** y α es a su vez una **Fórmula**.

- t1 = t2

Fórmulas

Una fórmula puede tener la siguiente forma, donde t_1 y t_2 son **Términos**, **at** son **Términos de Acción** y α es a su vez una **Fórmula**.

- t1 = t2
- $[]\alpha, [at]\alpha$

Fórmulas

Una fórmula puede tener la siguiente forma, donde t_1 y t_2 son **Términos**, **at** son **Términos de Acción** y α es a su vez una **Fórmula**.

- t1 = t2
- $[]\alpha, [at]\alpha$
- Cualquier combinación de estas fórmulas usando los conectivos y cuantificadores de la lógica de primer Orden.

Fórmulas

Una fórmula puede tener la siguiente forma, donde t_1 y t_2 son **Términos**, **at** son **Términos de Acción** y α es a su vez una **Fórmula**.

- t1 = t2
- $[]\alpha, [at]\alpha$
- Cualquier combinación de estas fórmulas usando los conectivos y cuantificadores de la lógica de primer Orden.

- **Términos** son variables, constantes, atributos de aridad n. $A(t_1, \ldots, t_n)$ o funciones $f(t_1, \ldots, t_n)$, donde $t_1, \ldots, t_n$ son a su vez términos.

Fórmulas

Una fórmula puede tener la siguiente forma, donde t_1 y t_2 son **Términos**, **at** son **Términos de Acción** y α es a su vez una **Fórmula**.

- t1 = t2
- $[]\alpha, [at]\alpha$
- Cualquier combinación de estas fórmulas usando los conectivos y cuantificadores de la lógica de primer Orden.

- **Términos** son variables, constantes, atributos de aridad n. $A(t_1, \ldots, t_n)$ o funciones $f(t_1, \ldots, t_n)$, donde $t_1, \ldots, t_n$ son a su vez términos.
- **Términos de Acción** tienen la forma:
  - $\top$ ó $\bot$.

Fórmulas

Una fórmula puede tener la siguiente forma, donde t_1 y t_2 son **Términos**, **at** son **Términos de Acción** y α es a su vez una **Fórmula**.

- t1 = t2
- $[]\alpha, [at]\alpha$
- Cualquier combinación de estas fórmulas usando los conectivos y cuantificadores de la lógica de primer Orden.

- **Términos** son variables, constantes, atributos de aridad n. $A(t_1, \ldots, t_n)$ o funciones $f(t_1, \ldots, t_n)$, donde $t_1, \ldots, t_n$ son a su vez términos.
- **Términos de Acción** tienen la forma:
  - $\top$ ó $\bot$.
  - $a(t_1, \ldots, t_n)$

Fórmulas

Una fórmula puede tener la siguiente forma, donde t_1 y t_2 son **Términos**, **at** son **Términos de Acción** y α es a su vez una **Fórmula**.

- t1 = t2
- $[]\alpha, [at]\alpha$
- Cualquier combinación de estas fórmulas usando los conectivos y cuantificadores de la lógica de primer Orden.

- **Términos** son variables, constantes, atributos de aridad n. $A(t_1, \ldots, t_n)$ o funciones $f(t_1, \ldots, t_n)$, donde $t_1, \ldots, t_n$ son a su vez términos.
- **Términos de Acción** tienen la forma:
  - $\top$ ó $\bot$.
  - $a(t_1, \ldots, t_n)$
  - $(at_1||at_2)$, $(at_1 + at_2)$ ó $\overline{at}$.

Fórmulas

Una fórmula puede tener la siguiente forma, donde t_1 y t_2 son **Términos**, **at** son **Términos de Acción** y α es a su vez una **Fórmula**.

- t1 = t2
- $[]\alpha, [at]\alpha$
- Cualquier combinación de estas fórmulas usando los conectivos y cuantificadores de la lógica de primer Orden.

- **Términos** son variables, constantes, atributos de aridad n. $A(t_1, \ldots, t_n)$ o funciones $f(t_1, \ldots, t_n)$, donde $t_1, \ldots, t_n$ son a su vez términos.
- **Términos de Acción** tienen la forma:
  - $\top$ ó $\bot$.
  - $a(t_1, \ldots, t_n)$
  - $(at_1||at_2)$, $(at_1 + at_2)$ ó $\overline{at}$.

Las acciones son interpretadas como funciones las cuales definen para cada estado el conjunto de eventos que realiza la misma.

Un Ejemplo Interesante

Biblioteca

Consideremos una **Biblioteca** muy simple, en donde los usuarios pueden **retirar** un libro, **devolverlo** o **renovar** el período de préstamo.

Algunos requerimientos importantes de esta biblioteca son:

- Un usuario debe **devolver** el libro dentro de los 21 días de préstamo.

Un Ejemplo Interesante

Biblioteca

Consideremos una **Biblioteca** muy simple, en donde los usuarios pueden **retirar** un libro, **devolverlo** o **renovar** el período de préstamo.

Algunos requerimientos importantes de esta biblioteca son:

- Un usuario debe **devolver** el libro dentro de los 21 días de préstamo.
- Si un usuario **renueva** el préstamo, el período de préstamo es extendido 21 días desde la fecha en que realiza la renovación.

Un Ejemplo Interesante

Biblioteca

Consideremos una **Biblioteca** muy simple, en donde los usuarios pueden **retirar** un libro, **devolverlo** o **renovar** el período de préstamo.

Algunos requerimientos importantes de esta biblioteca son:

- Un usuario debe **devolver** el libro dentro de los 21 días de préstamo.
- Si un usuario **renueva** el préstamo, el período de préstamo es extendido 21 días desde la fecha en que realiza la renovación.
- Si un usuario **"falla"** en devolver el libro dentro de los 21 días de préstamo:
  - Se le realiza una **multa** y se le extiende el préstamo por 7 días mas.

Un Ejemplo Interesante

Biblioteca

Consideremos una **Biblioteca** muy simple, en donde los usuarios pueden **retirar** un libro, **devolverlo** o **renovar** el período de préstamo.

Algunos requerimientos importantes de esta biblioteca son:

- Un usuario debe **devolver** el libro dentro de los 21 días de préstamo.
- Si un usuario **renueva** el préstamo, el período de préstamo es extendido 21 días desde la fecha en que realiza la renovación.
- Si un usuario **"falla"** en devolver el libro dentro de los 21 días de préstamo:
  - Se le realiza una **multa** y se le extiende el préstamo por 7 días mas.
- Sino no paga la **multa** dentro de los 7 días:
  - Se **bloquean** los permisos de préstamo de libros de dicho usuario y se incrementa la multa.

Una posible *Signature* para mi Biblioteca:

```
agent Library

Sorts          Variables          Constants
  BOOK           u: USER            books
  USER           b: BOOK            users
  BOOL                              fine_val:50p
  NAT
```

Una posible *Signature* para mi Biblioteca:

agent Library

Sorts	Variables	Constants
BOOK	u: USER	books
USER	b: BOOK	users
BOOL		fine_val:50p
NAT		

Attributes

- may_borrow: USERS $\longrightarrow$ BOOL
- has: BOOK $\longrightarrow$ USERS $\cup$ {*library*}
- due: BOOK $\longrightarrow$ NAT
- fine: USER x BOOK $\longrightarrow$ NAT
- fine_due: USER x BOOK $\longrightarrow$ NAT

Una posible *Signature* para mi Biblioteca:

agent Library

Sorts	Variables	Constants
BOOK	u: USER	books
USER	b: BOOK	users
BOOL		fine_val:50p
NAT		

Attributes
- may_borrow: USERS $\longrightarrow$ BOOL
- has: BOOK $\longrightarrow$ USERS $\cup$ {*library*}
- due: BOOK $\longrightarrow$ NAT
- fine: USER x BOOK $\longrightarrow$ NAT
- fine_due: USER x BOOK $\longrightarrow$ NAT

Actions borrow: USER x BOOK
- return: USER x BOOK
- renew: USER x BOOK
- block: USER
- unblock: USER
- pay: USER x BOOK x NAT
- issue_fine: USER x BOOK

Algunas propiedades deseables para mi Biblioteca(Theory)

Especificando Comportamiento:

```
(L1) []has(b) = library
(L2) [borrow(u,b)]has(b) = u
(L3) [return(u,b)]has(b) = library
```

Algunas propiedades deseables para mi Biblioteca(Theory)

Especificando Comportamiento:

(L1) []has(b) = library
(L2) [borrow(u,b)]has(b) = u
(L3) [return(u,b)]has(b) = library

Especificando algunas Restricciones:

(L4) PER(borrow(u,b)) $\leftrightarrow$ (has(b) = library $\wedge$ may_borrow(u) = true)
(L5) PER(return(u,b)) $\leftrightarrow$ (has(b) = u)
(L6) PER(renew(u,b)) $\leftrightarrow$ (has(b) = u $\wedge$ may_borrow(u) = true)

Algunas propiedades deseables para mi Biblioteca(Theory)

Especificando Comportamiento:

(L1) []has(b) = library
(L2) [borrow(u,b)]has(b) = u
(L3) [return(u,b)]has(b) = library

Especificando algunas Restricciones:

(L4) PER(borrow(u,b)) $\leftrightarrow$ (has(b) = library $\wedge$ may_borrow(u) = true)
(L5) PER(return(u,b)) $\leftrightarrow$ (has(b) = u)
(L6) PER(renew(u,b)) $\leftrightarrow$ (has(b) = u $\wedge$ may_borrow(u) = true)

Especificando Recuperación de Errores: *"Los derechos de préstamo serán devueltos sólo cuando pague todas las multas y devuelva todos los libros"*

(L19) ($\forall\bullet$ (fine(u,b) $\leq 0 \wedge$ has(b) $\neq$ u) $\wedge$ may_borrow(u)= false)$\leftrightarrow$ OBL(unblock(u))
(L20) [unblock(u)]may_borrow(u) = true

Lógicas Temporales

Las lógicas temporales son variantes de la lógica modal que se dedican a razonar sobre la relación temporal de eventos. Existen muchos tipos de lógicas temporales. Por Ejemplo:

- Lógica Temporal Lineal(LTL),

Lógicas Temporales

Las lógicas temporales son variantes de la lógica modal que se dedican a razonar sobre la relación temporal de eventos. Existen muchos tipos de lógicas temporales. Por Ejemplo:

- Lógica Temporal Lineal(LTL),
- Lógica de computaciones ramificadas(CTL),

Lógicas Temporales

Las lógicas temporales son variantes de la lógica modal que se dedican a razonar sobre la relación temporal de eventos. Existen muchos tipos de lógicas temporales. Por Ejemplo:

- Lógica Temporal Lineal(LTL),
- Lógica de computaciones ramificadas(CTL),
- etc.

Lógicas Temporales

Las lógicas temporales son variantes de la lógica modal que se dedican a razonar sobre la relación temporal de eventos. Existen muchos tipos de lógicas temporales. Por Ejemplo:

- Lógica Temporal Lineal(LTL),
- Lógica de computaciones ramificadas(CTL),
- etc.

$pCTL^*$

Aziz et al. definen una variante probabilística de CTL denominada $pCTL^*$, la cual permite cuantificar propiedades probabilísticas sobre *Sistemas Estocásticos*, es decir sistemas en los cuales hay una cierta probabilidad asociada con los eventos. Utilizan modelos probabilisticos para dar semántica a esta lógica.

Trabajos Futuros

La idea para estos próximos meses es tratar de combinar algunas de las ventajas que nos proveen los frameworks lógicos vistos tratando de incluir la noción de Probabilidades.

Ejemplo

Poder expresar propiedades como:

- "En el 86% de los casos el sensor de la bomba de insulina se recupera dentro del limite de tiempo permitido.".

Trabajos Futuros

La idea para estos próximos meses es tratar de combinar algunas de las ventajas que nos proveen los frameworks lógicos vistos tratando de incluir la noción de Probabilidades.

Ejemplo

Poder expresar propiedades como:

- "En el 86% de los casos el sensor de la bomba de insulina se recupera dentro del limite de tiempo permitido.".
- "Hay un 60% de probabilidad que los usuarios paguen la multa y devuelvan los libros.".

Otra motivación que tenemos es investigar si tiene sentido considerar el "pasado" a la hora de calcular las probabilidades.

Bibliografía

[Lamport] L. Lamport, R.E. Shostak and M.C. Pease. The Byzantine Generals Problem. ACM Trans. Program. Lang. Syst. 4(3): 382-401 (1982)
[Fiadeiro & Maibaum] J.L. Fiadeiro and T.S.E. Maibaum. Temporal reasoning over deontic specifications. J. Log. Comput. 1(3): 357-395 (1991)
[Kent & Maibaum & Quirk] S.J.H. Kent, T.S.E. Maibaum and W.J. Quirk. Formally specifying temporal constraints and error recovery. In Proceedings of the 1st IEEE International Symposium on Requirements Engineering, IEEE CS Press, 1993, 208–215.
[Aziz] A. Aziz, V. Singhal, R.K. Brayton and A.L. Sangiovanni-Vincentelli. It Usually Works: The Temporal Logic of Stochastic Systems. Computer Aided Verification, 7th International Conference, Liège, Belgium, July, 3-5, 1995, Proceedings 1995.