# Modified tableaux for some kinds of multi-modal logics

### Emilio Gómez-Caminero, Ángel Nepomuceno Grupo de Lógica, Lenguaje e Información Universidad de Sevilla

Lisbon, 2012

Emilio Gómez-Caminero, Ángel Nepomuceno Modified tableaux for some kinds of multi-modal logics

• It is well known that we can use a labelled tableaux method like a decision procedure in modal logic.

→ Ξ →

- It is well known that we can use a labelled tableaux method like a decision procedure in modal logic.
- We need a system of labels of the form 1, 1.1, 1.2, 1.2.1, and so on.

- It is well known that we can use a labelled tableaux method like a decision procedure in modal logic.
- We need a system of labels of the form 1, 1.1, 1.2, 1.2.1, and so on.
- The recursive definition is:
  - a. 1 is a label. b. If  $\sigma$  is a label, then  $\sigma.n$  is a label (for  $n \ge 1$ )

- It is well known that we can use a labelled tableaux method like a decision procedure in modal logic.
- We need a system of labels of the form 1, 1.1, 1.2, 1.2.1, and so on.
- The recursive definition is:
  - a. 1 is a label.
  - b. If  $\sigma$  is a label, then  $\sigma . n$  is a label (for  $n \ge 1$ )
- Intuitively, each label represents a possible world, such that  $\sigma.n$  is reachable from  $\sigma$ .

• The rules for the propositional operators are the usual, whit a label which remains invariable. E.g.:

$$\frac{\sigma :: \alpha \land \beta}{\sigma :: \alpha}$$

→ 3 → 4 3

• The rules for the propositional operators are the usual, whit a label which remains invariable. E.g.:

$$\frac{\sigma :: \alpha \land \beta}{\sigma :: \alpha} \\ \sigma :: \beta$$

The rule for the operator 
 \u03c6 is the only one which creates a
 new label:

$$\mathbf{R}\diamondsuit:\frac{\sigma::\diamondsuit\alpha}{\sigma.n::\alpha}$$

(Where *n* is the first positive integer such that  $\sigma$ .*n* is new in the branch.)

The preceding rule, in addition whit the rules for the operator
 □ may give rise to an infinite branch.

- The preceding rule, in addition whit the rules for the operator
   □ may give rise to an infinite branch.
- But we can avoid it using the following restriction:

Except if  $\tau :: \alpha$  appears in the branch and  $\sigma$  is reachable since  $\tau$ . In that case, the rule is considered as applied and the formula is marked.

• The rule for the operator  $\Box$  depends on one of the the properties of the accessibility relation:

- The rule for the operator  $\Box$  depends on one of the the properties of the accessibility relation:
- If the accessibility relation is reflexive, the rule is:

$$\mathbf{R}\Box \text{ refl.}: \frac{\sigma :: \Box \alpha}{\sigma :: \alpha}$$

- The rule for the operator  $\Box$  depends on one of the the properties of the accessibility relation:
- If the accessibility relation is reflexive, the rule is:

$$\mathbf{R}\Box \text{ refl.}: \frac{\sigma :: \Box \alpha}{\sigma :: \alpha}$$

• If the accessibility relation is serial, the rule is:

$$\mathbf{R}\square \mathbf{ser.} : \frac{\sigma :: \square \alpha}{\sigma :: \Diamond \alpha}$$

• The specific character of the operator  $\Box$  is captured by the so called *inheritance rules*:

- The specific character of the operator  $\Box$  is captured by the so called *inheritance rules*:
- Basic case (the accessibility relation has no more properties than reflexivity or seriality):

$$\mathbf{IRT}: \frac{\sigma::\Box\alpha}{\sigma.n::\alpha}$$

(For any label  $\sigma$ .*n* which appears in the branch.)

We can introduce additional properties replacing the preceding rule whit the followings:

• Euclidianity (S4, KD4):

**IRS4** : 
$$\frac{\sigma :: \Box \alpha}{\sigma . n :: \Box \alpha}$$

We can introduce additional properties replacing the preceding rule whit the followings:

• Euclidianity (S4, KD4):

**IRS4** : 
$$\frac{\sigma :: \Box \alpha}{\sigma . n :: \Box \alpha}$$

• Symmetry (S5, KD45):

$$\mathsf{IRS5}: = \frac{\sigma :: \Box \alpha}{\sigma.n :: \Box \alpha}$$

- With the adequate combination of these rules we can create tableau methods for all basic systems of alethic modal logic.
- We can prove that these methods are correct and complete.
- In many cases, we can extend this kind of labelled tableau to systems of multi-modal logic.

• The easier extension of modal logic is the case in which we have a certain number of modal operators of the same kind.

- The easier extension of modal logic is the case in which we have a certain number of modal operators of the same kind.
- This is the case of multi-agent modal logics. The best known are epistemic and doxastic logic.

- The easier extension of modal logic is the case in which we have a certain number of modal operators of the same kind.
- This is the case of multi-agent modal logics. The best known are epistemic and doxastic logic.
- Given a set  $\mathcal{A}$  of agents, we have an operator  $\Box_{a_i}$  and its dual  $\Diamond_{a_i}$  for each agent  $a_i \in \mathcal{A}$ .

- The easier extension of modal logic is the case in which we have a certain number of modal operators of the same kind.
- This is the case of multi-agent modal logics. The best known are epistemic and doxastic logic.
- Given a set  $\mathcal{A}$  of agents, we have an operator  $\Box_{a_i}$  and its dual  $\Diamond_{a_i}$  for each agent  $a_i \in \mathcal{A}$ .
- Usually we write  $K_{a_i}$  when we are talking about epistemic logic and  $B_{a_i}$  when we are dealing with doxastic logic.

To adapt the tableau method to multi-agent modal logic we only have to adapt the form of the labels:

labels
Given a set $\mathcal A$ of agents:
a 1 is a label.
b If $\sigma$ is a label, then $\sigma.a_in$ is a label too (for $n \ge 1$ and $a_i \in A$ ).

The rules are the same that in the general case, but adapted to the new labels:

• 
$$\Diamond$$
-Rule (R $\Diamond$ ):  
 $\frac{\sigma :: \Diamond_{a_i} \alpha}{\sigma. a_i n :: \alpha}$   
•  $\Box$ -Rule (R $\Box$ ):

Knowledge	Belief
$\frac{\sigma::\Box_{\mathbf{a}_{i}}\alpha}{\sigma::\alpha}$	$\frac{\sigma::\Box_{\mathbf{a}_i}\alpha}{\sigma::\diamondsuit_{\mathbf{a}_i}\alpha}$

$$T_m/KD_m: \frac{\sigma :: \Box_{a_i}\alpha}{\sigma \cdot a_i n :: \alpha}$$

$$S4_m/KD4_m: \frac{\sigma :: \Box_{a_i}\alpha}{\sigma \cdot a_i n :: \Box \alpha}$$

$$S5_m/KD5_m: \frac{\sigma :: \Box_{a_i}\alpha}{\sigma \cdot a_i n :: \Box \alpha}$$

Emilio Gómez-Caminero, Ángel Nepomuceno Modified tableaux for some kinds of multi-modal logics

・聞き ・ ヨキ・ ・ ヨキ

æ

## Combining modalities

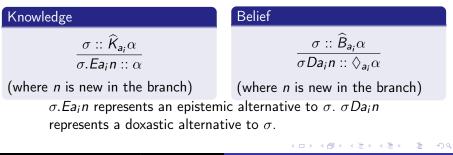
• We may combine different kinds of modalities; for example, epistemic and doxastic modalities, epistemic and deontic modalities, and son on.

## Combining modalities

- We may combine different kinds of modalities; for example, epistemic and doxastic modalities, epistemic and deontic modalities, and son on.
- In order to do it, we have to consider different kinds of accessibility relations and represent them using the labels.

## Combining modalities

- We may combine different kinds of modalities; for example, epistemic and doxastic modalities, epistemic and deontic modalities, and son on.
- In order to do it, we have to consider different kinds of accessibility relations and represent them using the labels.
- For example, if we want to combine epistemic and doxastic operators, rules are:



## Combining modalities. Inheritance rules.

• We have to consider the relations between different kinds of modality.

- We have to consider the relations between different kinds of modality.
- Depending on these relations, we have to change the inheritance rules. For example, if we accept that  $K_{a_i}\varphi \rightarrow B_{a_i}\varphi$ , we have to modify the inheritance rule for K (we give the example for S4):

$$\frac{\sigma :: K_{a_i}\alpha}{\sigma.Xa_in :: \alpha}$$

(Where X may be E or D)

• We can extend our multi-agent epistemic/doxastic logic (¿and deontic,...?) with group knowledge (group belief) operators.

- We can extend our multi-agent epistemic/doxastic logic (¿and deontic,...?) with group knowledge (group belief) operators.
- Eφ means everybody knows (believes) that φ. It can be defined as □<sub>a1</sub>φ ∧ □<sub>a2</sub>φ ∧ ··· (for any a<sub>i</sub> ∈ A).

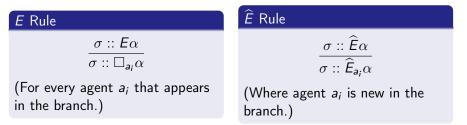
- We can extend our multi-agent epistemic/doxastic logic (¿and deontic,...?) with group knowledge (group belief) operators.
- Eφ means everybody knows (believes) that φ. It can be defined as □<sub>a1</sub>φ ∧ □<sub>a2</sub>φ ∧ ··· (for any a<sub>i</sub> ∈ A).
- Its dual, *Ê*φ, means *it is possible for somebody that* φ. It can be defined as ◊<sub>a1</sub>φ ∨ ◊<sub>a2</sub>φ ∨ · · · (for any a<sub>i</sub> ∈ A).

- We can extend our multi-agent epistemic/doxastic logic (¿and deontic,...?) with group knowledge (group belief) operators.
- Eφ means everybody knows (believes) that φ. It can be defined as □<sub>a1</sub>φ ∧ □<sub>a2</sub>φ ∧ ··· (for any a<sub>i</sub> ∈ A).
- Its dual, *Ê*φ, means *it is possible for somebody that* φ. It can be defined as ◊<sub>a1</sub>φ ∨ ◊<sub>a2</sub>φ ∨ · · · (for any a<sub>i</sub> ∈ A).
- $C\varphi$  means it is common knowledge (belief) that  $\varphi$ . It can be intuitively understood as the infinite conjunction  $E\varphi \wedge EE\varphi \wedge EEE\varphi \wedge \cdots$

・ 同 ト ・ ヨ ト ・ ヨ ト …

- We can extend our multi-agent epistemic/doxastic logic (¿and deontic,...?) with group knowledge (group belief) operators.
- Eφ means everybody knows (believes) that φ. It can be defined as □<sub>a1</sub>φ ∧ □<sub>a2</sub>φ ∧ ··· (for any a<sub>i</sub> ∈ A).
- Its dual, *Ê*φ, means *it is possible for somebody that* φ. It can be defined as ◊<sub>a1</sub>φ ∨ ◊<sub>a2</sub>φ ∨ · · · (for any a<sub>i</sub> ∈ A).
- Cφ means it is common knowledge (belief) that φ. It can be intuitively understood as the infinite conjunction Eφ ∧ EEφ ∧ EEEφ ∧ ···
- Its dual, C
   *φ* (it is compatible with common knowledge (belief) that φ) can be intuitively understood as the infinite disjunction E
   *φ* ∨ E
   *Ê*E
   *φ* ∨ ···

# E and $\hat{E}$ may be treated as quantifiers:



#### *C* is treated in the same way that the operator $\Box$ :

Knowledge	Belief
$\frac{\sigma:: C\alpha}{\sigma:: \alpha}$	$\frac{\sigma :: \mathcal{C}\alpha}{\sigma :: \widehat{\mathcal{C}}\alpha}$

伺 ト イヨト イヨ

IRC:  
IRT: 
$$\frac{\sigma :: C\alpha}{\sigma.a_i n :: C\alpha}$$
  
(For any label  $\sigma.a_i n$  that appears in the branch)  
IRS5:  $\frac{\sigma :: C\alpha}{\sigma.a_i n :: C\alpha}$   
(For any label  $\sigma, \sigma.a_i n$  that appears in the branch)

The case of S4 is like T, but we have to consider the cases where we have applied the restriction of the rule  $R\Diamond$ .

• Before talking about the operator  $\widehat{C}$ , we have to speak about DB-tableaux.

- Before talking about the operator  $\widehat{C}$ , we have to speak about DB-tableaux.
- We call DB-tableaux to a modified tableau method proposed independently by Díaz Estévez and Boolos to deal with formulas of the form ∀x∃yφ(x).

- Before talking about the operator  $\widehat{C}$ , we have to speak about DB-tableaux.
- We call DB-tableaux to a modified tableau method proposed independently by Díaz Estévez and Boolos to deal with formulas of the form ∀x∃yφ(x).
- In this method, the rule

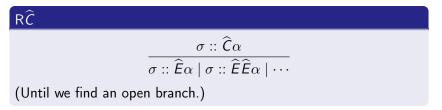
$$\frac{\exists x\varphi}{\varphi\left(k_{n+1}/x\right)}$$

(where  $k_n$  is the last parameter that appears in the branch) is replaced with:

$$\frac{\exists x\varphi}{\varphi(k_1/x) \mid \cdots \mid \varphi(k_n/x) \mid \varphi(k_{n+1}/x)}$$

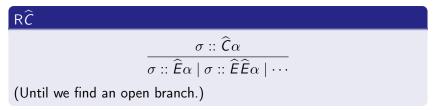


• We will use an infinitary version of the previous rule to deal with the operator  $\widehat{C}$ .





• We will use an infinitary version of the previous rule to deal with the operator  $\widehat{C}$ .



• If the formula has a model, we will find it in a finite number of steps; if the formula has not a model, the tableau becomes infinite.

- $\bullet$  We can interpret  $\Box$  and  $\diamondsuit$  as temporal operators:
  - $\Box \varphi$  means "always (now and in the future)  $\varphi$ ".
  - $\Diamond \varphi$  means "sometime (now or in the future)  $\varphi$ ".

- We can interpret  $\Box$  and  $\Diamond$  as temporal operators:
  - $\Box \varphi$  means "always (now and in the future)  $\varphi$ ".
  - $\Diamond \varphi$  means "sometime (now or in the future)  $\varphi$ ".
- It is usual to introduce two more operators:  $\bigcirc$  y U (we are not going to deal with branching-time operators):
  - $\bigcirc \varphi$  means " at the next moment,  $\varphi$  ".
  - $\varphi U \psi$  means " $\varphi$  until  $\psi$ ".

We have to introduce the following changes:

Labels: Each label  $t \in \mathbb{N}$  represents a moment of the time (we are dealing with discrete time).

We have to introduce the following changes:

Labels: Each label  $t \in \mathbb{N}$  represents a moment of the time (we are dealing with discrete time).

Operator  $\Box$ : Rules for  $\Box$  are similar to the previous cases:



#### Operator $\bigcirc$ : The rules for $\bigcirc$ and its negation are:



▲ □ ▶ ▲ □ ▶ ▲ □ ▶

э

• For dealing with the operators  $\Diamond$  and U (and it negation) we need *recursive rules*.

- For dealing with the operators  $\Diamond$  and U (and it negation) we need *recursive rules*.
- Recursive rules are of the form

$$\frac{A}{SC \parallel RC}$$

- For dealing with the operators  $\Diamond$  and U (and it negation) we need *recursive rules*.
- Recursive rules are of the form

$$\frac{A}{SC \parallel RC}$$

• SC is the stop condition. RC is the recursive clause.

- For dealing with the operators  $\Diamond$  and U (and it negation) we need *recursive rules*.
- Recursive rules are of the form

$$\frac{A}{SC \parallel RC}$$

- SC is the stop condition. RC is the recursive clause.
- If SC gives rise to an open branch, we have finished the application of the rule.

- For dealing with the operators  $\Diamond$  and U (and it negation) we need *recursive rules*.
- Recursive rules are of the form

$$\frac{A}{SC \parallel RC}$$

- SC is the stop condition. RC is the recursive clause.
- If SC gives rise to an open branch, we have finished the application of the rule.
- If SC gives rise to a closed branch, we have to apply RC, which makes us to apply the rule again at the next moment of the time.

- For dealing with the operators  $\Diamond$  and U (and it negation) we need *recursive rules*.
- Recursive rules are of the form

$$\frac{A}{SC \parallel RC}$$

- SC is the *stop condition*. RC is the *recursive clause*.
- If SC gives rise to an open branch, we have finished the application of the rule.
- If SC gives rise to a closed branch, we have to apply RC, which makes us to apply the rule again at the next moment of the time.
- One more time, if the formula has a finite model, we find it in a finite number of steps; if the formula has not a model, the tableau becomes infinite.

• R◊:  $\frac{t::\Diamond\alpha}{t::\alpha\parallel t::\bigcirc\Diamond\alpha}$ • RU:  $\frac{\beta U\alpha}{t :: \beta \left\| \begin{array}{c} t :: \alpha \\ t :: \cap \alpha U\beta \end{array} \right\|}$ •  $\mathbf{R} \neg U$ :  $t :: \neg (\alpha U \beta)$  $\begin{array}{c} t ::: \neg \alpha \\ t ::: \neg \beta \end{array} \middle\| \begin{array}{c} \hline t ::: \alpha \\ t ::: \neg \\ t ::: \neg \end{array} \\ t :: \bigcirc \neg (\alpha U \beta) \end{array}$ 

< /₽ > < E > <

• We can present labelled tableau methods for modal logic. The so called *inheritance rules* expresses the properties of the accessibility relation. We can prove that these methods are correct and complete.

- We can present labelled tableau methods for modal logic. The so called *inheritance rules* expresses the properties of the accessibility relation. We can prove that these methods are correct and complete.
- We can extend these methods to multi-modal logics using more complicated labels and rules.

- We can present labelled tableau methods for modal logic. The so called *inheritance rules* expresses the properties of the accessibility relation. We can prove that these methods are correct and complete.
- We can extend these methods to multi-modal logics using more complicated labels and rules.
- For some infinitary operators we have to use DB-Tableaux.

- We can present labelled tableau methods for modal logic. The so called *inheritance rules* expresses the properties of the accessibility relation. We can prove that these methods are correct and complete.
- We can extend these methods to multi-modal logics using more complicated labels and rules.
- For some infinitary operators we have to use DB-Tableaux.
- For temporal operators we use recursive rules.

- We can present labelled tableau methods for modal logic. The so called *inheritance rules* expresses the properties of the accessibility relation. We can prove that these methods are correct and complete.
- We can extend these methods to multi-modal logics using more complicated labels and rules.
- For some infinitary operators we have to use DB-Tableaux.
- For temporal operators we use recursive rules.
- In the last two cases, the tableau may become infinite.

# Thank you Muito obrigado

Emilio Gómez-Caminero, Ángel Nepomuceno Modified tableaux for some kinds of multi-modal logics