

# Autoroute

Generated by Doxygen 1.8.6

Thu Feb 12 2015 16:56:24



# Contents

<b>1</b>	<b>Documentation du logiciel Autoroute</b>	<b>1</b>
1.1	Introduction	1
1.2	Installation du logiciel pour les utilisateurs	1
1.3	Pour les développeurs	1
1.3.1	Etape 1 : Prise en main des sources et execution	1
1.3.2	Etape 2 : Arborescence du projet	2
1.4	Définitions	2
1.4.1	Minimisation d'un automate	2
1.4.2	Standardisation d'un automate	2
1.4.3	Produit de deux automates	2
1.4.4	Détermination de deux automates	2
<b>2</b>	<b>Hierarchical Index</b>	<b>3</b>
2.1	Class Hierarchy	3
<b>3</b>	<b>Class Index</b>	<b>5</b>
3.1	Class List	5
<b>4</b>	<b>File Index</b>	<b>7</b>
4.1	File List	7
<b>5</b>	<b>Class Documentation</b>	<b>9</b>
5.1	Automate Class Reference	9
5.1.1	Constructor & Destructor Documentation	10
5.1.1.1	Automate	10
5.1.1.2	Automate	10
5.1.1.3	~Automate	10
5.1.2	Member Function Documentation	10
5.1.2.1	ajoutEtat	10
5.1.2.2	ajoutTransition	11
5.1.2.3	cible_transition	12
5.1.2.4	determinise	12
5.1.2.5	getAlpha	12

5.1.2.6	<a href="#">getEtat</a>	12
5.1.2.7	<a href="#">getEtats</a>	13
5.1.2.8	<a href="#">getNbTransition</a>	13
5.1.2.9	<a href="#">getTabTransitions</a>	13
5.1.2.10	<a href="#">isDeterministe</a>	13
5.1.2.11	<a href="#">isStandard</a>	13
5.1.2.12	<a href="#">minimise</a>	14
5.1.2.13	<a href="#">produit</a>	14
5.1.2.14	<a href="#">standardise</a>	14
5.1.2.15	<a href="#">supprimeEtat</a>	14
5.1.2.16	<a href="#">supprimeEtat</a>	14
5.1.2.17	<a href="#">supprimerEtatsNonAccessibles</a>	15
5.1.2.18	<a href="#">toDot</a>	15
5.1.3	<a href="#">Member Data Documentation</a>	15
5.1.3.1	<a href="#">etats</a>	15
5.2	<a href="#">choixPointe Class Reference</a>	15
5.2.1	<a href="#">Constructor &amp; Destructor Documentation</a>	16
5.2.1.1	<a href="#">choixPointe</a>	16
5.2.2	<a href="#">Member Function Documentation</a>	16
5.2.2.1	<a href="#">changeEvent</a>	16
5.2.2.2	<a href="#">resetAffichage</a>	16
5.2.2.3	<a href="#">sendDad</a>	16
5.3	<a href="#">CreateAutomate Class Reference</a>	17
5.3.1	<a href="#">Constructor &amp; Destructor Documentation</a>	18
5.3.1.1	<a href="#">CreateAutomate</a>	18
5.3.2	<a href="#">Member Function Documentation</a>	18
5.3.2.1	<a href="#">ajoutEtat</a>	18
5.3.2.2	<a href="#">changeState</a>	18
5.3.2.3	<a href="#">displayRight</a>	18
5.3.2.4	<a href="#">supprimeEtat</a>	18
5.3.3	<a href="#">Member Data Documentation</a>	19
5.3.3.1	<a href="#">a</a>	19
5.3.3.2	<a href="#">left</a>	19
5.3.3.3	<a href="#">maVue</a>	19
5.3.3.4	<a href="#">right</a>	19
5.4	<a href="#">etat Class Reference</a>	19
5.4.1	<a href="#">Constructor &amp; Destructor Documentation</a>	20
5.4.1.1	<a href="#">etat</a>	20
5.4.1.2	<a href="#">etat</a>	20
5.4.1.3	<a href="#">etat</a>	20

5.4.2	Member Function Documentation	21
5.4.2.1	ajoutTransition	21
5.4.2.2	estDansList	21
5.4.2.3	find_transition	21
5.4.2.4	find_transition	21
5.4.2.5	getName	22
5.4.2.6	getNameF	22
5.4.2.7	getNumber	22
5.4.2.8	getTransitions	22
5.4.2.9	isFinal	22
5.4.2.10	isInitial	22
5.4.2.11	operator!=	23
5.4.2.12	operator==	23
5.4.2.13	renameTransition	23
5.4.2.14	setFinal	23
5.4.2.15	setInitial	23
5.4.2.16	setName	24
5.4.2.17	setNumber	24
5.4.2.18	supprimeTransition	24
5.4.3	Member Data Documentation	24
5.4.3.1	numero	24
5.5	etatLeft Class Reference	24
5.5.1	Constructor & Destructor Documentation	25
5.5.1.1	etatLeft	25
5.5.2	Member Data Documentation	25
5.5.2.1	numero	25
5.6	etatRight Class Reference	26
5.6.1	Constructor & Destructor Documentation	26
5.6.1.1	etatRight	26
5.6.2	Member Function Documentation	27
5.6.2.1	addTransition	27
5.6.2.2	eraseTransition	27
5.6.3	Member Data Documentation	27
5.6.3.1	a	27
5.6.3.2	addTrans	27
5.6.3.3	numero	27
5.7	MainWindow Class Reference	27
5.7.1	Constructor & Destructor Documentation	28
5.7.1.1	MainWindow	28
5.7.1.2	~MainWindow	29

5.7.2	Member Function Documentation	29
5.7.2.1	creerAuto	29
5.7.2.2	getDetermin	29
5.7.2.3	getMinimisation	29
5.7.2.4	getPrecedent	29
5.7.2.5	getProduit	29
5.7.2.6	getStandard	29
5.7.2.7	getSuivant	29
5.7.2.8	info	30
5.7.2.9	openFile	30
5.7.2.10	resetUi	30
5.8	Transition Class Reference	30
5.8.1	Constructor & Destructor Documentation	31
5.8.1.1	Transition	31
5.8.1.2	~Transition	31
5.8.2	Member Data Documentation	31
5.8.2.1	cible	31
5.8.2.2	vocab	31
<b>6</b>	<b>File Documentation</b>	<b>33</b>
6.1	/home/aaiighht/Bureau/projetTL/automate-project/automate.h File Reference	33
6.1.1	Detailed Description	33
6.1.2	Function Documentation	33
6.1.2.1	equal	33
6.1.2.2	isFinal	34
6.2	/home/aaiighht/Bureau/projetTL/automate-project/choixpointe.h File Reference	34
6.2.1	Detailed Description	34
6.3	/home/aaiighht/Bureau/projetTL/automate-project/createautomate.h File Reference	34
6.3.1	Detailed Description	35
6.4	/home/aaiighht/Bureau/projetTL/automate-project/etatleft.h File Reference	35
6.4.1	Detailed Description	35
6.5	/home/aaiighht/Bureau/projetTL/automate-project/etatright.h File Reference	35
6.5.1	Detailed Description	35
6.6	/home/aaiighht/Bureau/projetTL/automate-project/mainwindow.h File Reference	35
6.6.1	Detailed Description	36
6.7	/home/aaiighht/Bureau/projetTL/automate-project/transition.h File Reference	36
6.7.1	Detailed Description	36
<b>Index</b>		<b>37</b>

# Chapter 1

## Documentation du logiciel Autoroute

### 1.1 Introduction

En allant dans la section Classes, vous auriez accès à la documentation de l'ensemble des classes. A partir de là, vous pourriez trouver de la doc concernant les attributs et méthodes des classes.

Dans cette page principale, nous verrons comment exécuter le logiciel Autoroute, prendre en main les sources, l'arborescence du projet ainsi que des définitions relatives à la théorie des langages et aux automates.

### 1.2 Installation du logiciel pour les utilisateurs

Bla Bla Bla

### 1.3 Pour les développeurs

#### 1.3.1 Etape 1 : Prise en main des sources et execution

Ce logiciel est développé en C++, avec le framework QT5. La manière la plus simple d'accéder aux sources, d'exécuter le programme et de modifier ce logiciel est la suivante :

- installer QT
- créer un dossier dans lequel vous mettrez les 3 dossiers (executable, doc et automate-project)
- dans QT, cliquez sur Open a project puis allez chercher le fichier automate-project/autoroute.pro
- pour lancer le logiciel, cliquez simplement sur la flèche verte

Il vous faudra peut-être configurer dans l'onglet "Projects" l'exécution. Il suffit normalement de préciser le dossier automate/project et d'utiliser les paramètres par défaut.

NB : Vous aurez peut-être un problème de version si vous avez une version supérieure à QT5. Il suffit en général de modifier le nom des bibliothèques. Si cela ne change pas, il vous reste plusieurs solutions :

- aller voir sur le net comment passer le projet de QT5 à la version actuelle de QT
- résoudre les erreurs de compilation (aidez vous du debugger de QT), c'est la solution conseillée.

### 1.3.2 Etape 2 : Arborescence du projet

- doc/ : vous trouverez ici deux dossiers (html et latex) correspondant à deux formats de la documentation.

Il est possible d'ouvrir ce fichier avec Doxygen et de générer la documentation du programme si vous voulez la modifier. Ce tutoriel est assez bien fait pour prendre en main doxygen : <http://franckh.developpez.com/tutoriels/outils/doxygen/>

- automate-project/ : les sources du programme.

Mieux vaut ne pas y toucher au début, surtout si l'on ne connaît pas QT et modifier le code seulement via QT.

- executable/ : tout les fichiers relatifs aux exécutables

## 1.4 Définitions

### 1.4.1 Minimisation d'un automate

### 1.4.2 Standardisation d'un automate

### 1.4.3 Produit de deux automates

### 1.4.4 Détermination de deux automates



## Chapter 2

# Hierarchical Index

### 2.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Automate . . . . .	9
etat . . . . .	19
QMainWindow	
CreateAutomate . . . . .	17
MainWindow . . . . .	27
QWidget	
choixPointe . . . . .	15
etatLeft . . . . .	24
etatRight . . . . .	26
Transition . . . . .	30



## Chapter 3

# Class Index

### 3.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

<a href="#">Automate</a>	9
<a href="#">choixPointe</a>	15
<a href="#">CreateAutomate</a>	17
<a href="#">etat</a>	19
<a href="#">etatLeft</a>	24
<a href="#">etatRight</a>	26
<a href="#">MainWindow</a>	27
<a href="#">Transition</a>	30



## Chapter 4

# File Index

### 4.1 File List

Here is a list of all documented files with brief descriptions:

/home/aaiighht/Bureau/projetTL/automate-project/ <a href="#">automate.h</a>	
Représente un automate, son seul attribut est un vector d'états . . . . .	33
/home/aaiighht/Bureau/projetTL/automate-project/ <a href="#">choixpointe.h</a>	
Utilisé pour la création d'un automate. Sert à définir les transitions en demandant à l'utilisateur de saisir vers quel état la transition va et quel étiquette porte cette transition . . . . .	34
/home/aaiighht/Bureau/projetTL/automate-project/ <a href="#">createautomate.h</a>	
Représente la fenêtre principale lors de la création d'un automate, contient les autres éléments : etatleft, etatright, transition et choixpointe . . . . .	34
/home/aaiighht/Bureau/projetTL/automate-project/ <a href="#">etat.h</a>	??
/home/aaiighht/Bureau/projetTL/automate-project/ <a href="#">etatleft.h</a>	
Représente la partie gauche lors de la création d'un automate, la partie listant les états . . . .	35
/home/aaiighht/Bureau/projetTL/automate-project/ <a href="#">etatright.h</a>	
Représente la partie sup droite lors de la création d'un automate, la partie listant les transitions d'un état, s'il est final/initial etc . . . . .	35
/home/aaiighht/Bureau/projetTL/automate-project/ <a href="#">mainwindow.h</a>	
Représente la fenetre principale du programme. On gère ici les listeners des boutons . . . . .	35
/home/aaiighht/Bureau/projetTL/automate-project/ <a href="#">transition.h</a>	
Représente une transition d'un état . . . . .	36



## Chapter 5

# Class Documentation

### 5.1 Automate Class Reference

#### Public Member Functions

- [Automate](#) ()  
*Constructeur sans paramètre.*
- [Automate](#) (const [Automate](#) &a)  
*Constructeur.*
- [~Automate](#) ()  
*Destructeur.*
- void [ajoutEtat](#) ([etat](#) cible)  
*Ajout d'un état.*
- vector< int > [getTabTransitions](#) ()  
*Liste les différentes transitions de l'automate.*
- bool [isDeterministe](#) ()  
*Test si un automate est déterministe.*
- bool [isStandard](#) ()  
*Test si un automate est standard.*
- [etat](#) \* [getEtat](#) (int number)  
*Retourne un pointeur vers l'état dont le numéro est précisé en paramètre.*
- vector< [etat](#) > [getEtats](#) ()  
*Retourne le vector d'états de l'automate.*
- void [ajoutTransition](#) ([etat](#) from, [etat](#) to, int vocab)  
*Ajoute une transition à un état.*
- void [supprimeEtat](#) ([etat](#) cible)  
*Supprime un état de l'automate.*
- int [cible\\_transition](#) (int etatDepart, int etiq)  
*Retourne l'état ciblé par une transition.*
- void [supprimeEtat](#) ([etat](#) cible, [Automate](#) \*a)  
*Supprime un état de l'automate passé en paramètre.*
- void [supprimerEtatsNonAccessibles](#) ([Automate](#) \*a)  
*Supprime les états non accessibles de l'automate passé en paramètre.*
- string [toDot](#) ()  
*Fonction permettant de renvoyer une chaîne à partir de l'automate actuel.*
- vector< [Automate](#) > [produit](#) ([Automate](#) A)  
*Réalise le produit de deux automates.*

- `vector< int > getAlpha ()`  
*Taille de l'alphabet de l'automate.*
- `vector< pair< Automate, string > > determinise ()`  
*Réalise la détermination de l'automate.*
- `vector< pair< Automate, string > > standardise ()`  
*Réalise la standardisation de l'automate.*
- `vector< pair< Automate, string > > minimise ()`  
*Réalise la minimisation de l'automate.*
- `int getNbTransition ()`  
*Renvoie le nombre de transition portant une étiquette différente.*

## Public Attributes

- `vector< etat > etats`

## 5.1.1 Constructor & Destructor Documentation

### 5.1.1.1 Automate::Automate ( )

Constructeur sans paramètre.

Constructeur de la classe [Automate](#), produit un automate vide.

### 5.1.1.2 Automate::Automate ( const Automate & a )

Constructeur.

Construit l'automate passé en paramètre

Parameters

<i>a</i>	: automate à construire
----------	-------------------------

### 5.1.1.3 Automate::~~Automate ( )

Destructeur.

Destructeur de la classe [Automate](#)

## 5.1.2 Member Function Documentation

### 5.1.2.1 void Automate::ajoutEtat ( etat cible )

Ajout d'un état.

Methode qui permet d'ajouter un état à l'automate

Parameters

<i>cible</i>	: l'état à ajouter
--------------	--------------------



### 5.1.2.2 void Automate::ajoutTransition ( *etat from*, *etat to*, int *vocab* )

Ajoute une transition à un état.

Ajoute une transition à l'état *from* en direction de l'état *to* et portant l'étiquette *vocab*.

**Parameters**

<i>from</i>	: l'état de départ de la transition
<i>to</i>	: l'état d'arrivée de la transition
<i>vocab</i>	: le numéro de la transition, son étiquette.

**5.1.2.3 int Automate::cible\_transition ( int *etatDepart*, int *etiq* )**

Retourne l'état ciblé par une transition.

Renvoie le numéro de l'état ciblé par la transition partant de *etatDepart* et portant l'étiquette *etiq*. Cette fonction ne fonctionne que si l'automate est déterministe, elle est utilisée seulement dans la minimisation

**Parameters**

<i>etatDepart</i>	: le numéro de l'état d'où part la transition
<i>etiq</i>	: l'étiquette de la transition, partant de <i>etatDepart</i>

**Returns**

le numéro de l'état ciblé par la transition, -1 s'il n'y en a pas

**5.1.2.4 vector< pair< Automate, string > > Automate::determinise ( )**

Réalise la détermination de l'automate.

Réalise la détermination de l'automate, et renvoie un vecteur. Dans ce vecteur, chaque élément correspond à une paire : un automate et une chaîne. Chaque élément pair représente en fait une étape dans le processus de détermination. La chaîne est le texte correspondant aux explications et l'automate est l'automate à afficher pendant cette étape.

**Returns**

le vecteur servant pour la détermination

**5.1.2.5 vector< int > Automate::getAlpha ( )**

Taille de l'alphabet de l'automate.

Fonction qui retourne la taille de l'alphabet d'un automate

**Returns**

un vecteur d'entier, représentant l'ensemble des étiquettes différentes des transitions, c'est-à-dire l'alphabet de l'automate

**5.1.2.6 etat \* Automate::getEtat ( int *number* )**

Retourne un pointeur vers l'état dont le numéro est précisé en paramètre.

Récupère le pointeur vers l'état dont le numéro est précisé en paramètre. Récupère cet état dans le vecteur d'états (un attribut de l'automate).

## Parameters

<i>number</i>	: le numéro de l'état à retourner
---------------	-----------------------------------

## Returns

Un pointeur vers l'état

**5.1.2.7** `vector< etat > Automate::getEtats ( )`

Retourne le vector d'états de l'automate.

## Returns

Le vecteur d'états de l'automate.

**5.1.2.8** `int Automate::getNbTransition ( )`

Renvoie le nombre de transition portant une étiquette différente.

Fonction utilisée dans la minimisation

## Returns

le nombre de transitions différentes de l'automate

**5.1.2.9** `vector< int > Automate::getTabTransitions ( )`

Liste les différentes transitions de l'automate.

Methode permettant de lister dans un vector d'int, les différentes transitions. [getTabTransitions\(\).size\(\)](#) permet donc de connaître le nombre de transitions différentes dans l'automate

## Returns

un vecteur, chaque entier du vecteur représentant un type de transition

**5.1.2.10** `bool Automate::isDeterministe ( )`

Test si un automate est déterministe.

Permet de tester si un automate est déterministe (voir définition d'un automate déterministe).

## Returns

true si l'automate est déterministe, false sinon

**5.1.2.11** `bool Automate::isStandard ( )`

Test si un automate est standard.

Permet de tester si un automate est standard (voir définition d'un automate standard).

## Returns

true si l'automate est standard, false sinon

#### 5.1.2.12 `vector< pair< Automate, string > > Automate::minimise ( )`

Réalise la minimisation de l'automate.

Réalise la minimisation de l'automate, et renvoie un vecteur. Dans ce vecteur, chaque élément correspond à une pair : un automate et une chaîne. Chaque élément pair représente en fait une étape dans le processus de minimisation. La chaîne est le texte correspondant aux explications et l'automate est l'automate à afficher pendant cette étape.

##### Returns

le vecteur servant pour la minimisation

#### 5.1.2.13 `vector< Automate > Automate::produit ( Automate A )`

Réalise le produit de deux automates.

Réalise le produit de deux automates (this et A), et renvoie un vecteur d'automates. Dans ce vecteur, chaque élément correspond à une étape du processus de produit de 2 automates.

##### Parameters

<i>A</i>	: le produit est réalisé avec cet automate A, passé en paramètre
----------	--

##### Returns

un vecteur d'automate, chaque automate correspondant à une étape dans le logiciel

#### 5.1.2.14 `vector< pair< Automate, string > > Automate::standardise ( )`

Réalise la standardisation de l'automate.

Réalise la standardisation de l'automate, et renvoie un vecteur. Dans ce vecteur, chaque élément correspond à une pair : un automate et une chaîne. Chaque élément pair représente en fait une étape dans le processus de standardisation. La chaîne est le texte correspondant aux explications et l'automate est l'automate à afficher pendant cette étape.

##### Returns

le vecteur servant pour la standardisation

#### 5.1.2.15 `void Automate::supprimeEtat ( etat cible )`

Supprime un état de l'automate.

Supprime l'état, passé en paramètre, de l'automate.

##### Parameters

<i>cible</i>	: l'état à supprimer
--------------	----------------------

#### 5.1.2.16 `void Automate::supprimeEtat ( etat cible, Automate * a )`

Supprime un état de l'automate passé en paramètre.

Supprime l'état cible de l'automate dont le pointeur a est passé en paramètre.

## Parameters

<i>cible</i>	: l'état à supprimer
<i>a</i>	: pointeur vers l'automate

## 5.1.2.17 void Automate::supprimerEtatsNonAccessibles ( Automate \* a )

Supprime les états non accessibles de l'automate passé en paramètre.

Supprime les états non accessibles de l'automate dont le pointeur est passé en paramètre

## Parameters

<i>a</i>	: pointeur vers l'automate
----------	----------------------------

## 5.1.2.18 string Automate::toDot ( )

Fonction permettant de renvoyer une chaine à partir de l'automate actuel.

Cette chaine correspond à la représentation de l'automate en graphe dans le langage de description DOT

## Returns

La chaine décrivant l'automate, à mettre dans un .dot ensuite

## 5.1.3 Member Data Documentation

## 5.1.3.1 vector&lt;etat&gt; Automate::etats

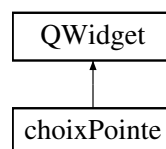
Vecteur des états de l'automate

The documentation for this class was generated from the following files:

- /home/aaiighht/Bureau/projetTL/automate-project/[automate.h](#)
- /home/aaiighht/Bureau/projetTL/automate-project/automate.cpp

## 5.2 choixPointe Class Reference

Inheritance diagram for choixPointe:



## Public Slots

- void [sendDad](#) ()

*Listener du bouton +.*

## Signals

- void `add` (int cible, int vocab)  
*Signal permettant d'ajouter la transition.*

## Public Member Functions

- `choixPointe` (QWidget \*parent=0)  
*Constructeur.*
- `~choixPointe` ()  
*Destructeur.*
- void `resetAffichage` (Automate a)  
*Clean du widget.*

## Protected Member Functions

- void `changeEvent` (QEvent \*e)  
*Change d'évènement.*

### 5.2.1 Constructor & Destructor Documentation

#### 5.2.1.1 `choixPointe::choixPointe ( QWidget * parent = 0 )`

Constructeur.

Instancie cette partie de la fenêtre.

### 5.2.2 Member Function Documentation

#### 5.2.2.1 `void choixPointe::changeEvent ( QEvent * e )` [protected]

Change d'évènement.

Parameters

<code>e</code>	: l'évènement pour lequel on doit changer.
----------------	--

#### 5.2.2.2 `void choixPointe::resetAffichage ( Automate a )`

Clean du widget.

Methode permettant de nettoyer le widget de ce qui a été ajouté et de remettre a jour la liste de choix pour les états cibles

Parameters

<code>a</code>	: l'état qu'on est en train de construire
----------------	---

#### 5.2.2.3 `void choixPointe::sendDad ( )` [slot]

Listener du bouton +.

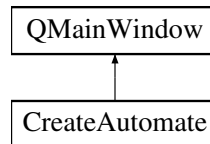
Méthode qui ajoute la transition seulement si les deux champs sont complétés. Listener du bouton +

The documentation for this class was generated from the following files:

- </home/aaiighht/Bureau/projetTL/automate-project/choixpointe.h>
- </home/aaiighht/Bureau/projetTL/automate-project/choixpointe.cpp>

## 5.3 CreateAutomate Class Reference

Inheritance diagram for CreateAutomate:



### Public Slots

- void [ajoutEtat](#) (bool ini=false, bool fina=false)  
*Ajoute un état à la fenetre.*
- void [afficherAutomate](#) ()  
*Affiche un automate en grand.*
- void [displayRight](#) (int toDisplay)  
*Affiche l'etat right spécifié*
- void [supprimeEtat](#) (int cible)  
*Supprime un état et réactualise l'affichage.*
- void [refreshAll](#) ()  
*Réactualise tout l'affichage.*
- void [sauvegarder](#) ()  
*Sauvegarder l'automate créé*
- void [changeState](#) (int [etat](#), bool ini, bool final)  
*Modifie un état.*

### Public Member Functions

- [CreateAutomate](#) (QWidget \*parent=0)  
*Constructeur.*
- [~CreateAutomate](#) ()  
*Destructeur.*
- void [resetAllListChoix](#) ()  
*Remet à jour la liste des états de droite.*
- void [displayAutomate](#) ()  
*Réactualise l'affichage de l'automate dans maVue.*

### Public Attributes

- [Automate](#) a
- QSvgWidget \* [maVue](#)
- int **actuel**
- vector< [etatLeft](#) \* > [left](#)
- vector< [etatRight](#) \* > [right](#)

## Protected Member Functions

- void **changeEvent** (QEvent \*e)
- void **adjust** ()

*Ajuste l'affichage.*

### 5.3.1 Constructor & Destructor Documentation

#### 5.3.1.1 CreateAutomate::CreateAutomate ( QWidget \* parent = 0 )

Constructeur.

Instancie cette partie de la fenêtre.

### 5.3.2 Member Function Documentation

#### 5.3.2.1 void CreateAutomate::ajoutEtat ( bool ini = false, bool fina = false ) [slot]

Ajoute un état à la fenetre.

Construit un etat left et right associé à ce nouvel état.

Parameters

<i>ini</i>	: true si l'état est initial
<i>fina</i>	: true si l'état est final

#### 5.3.2.2 void CreateAutomate::changeState ( int etat, bool ini, bool final ) [slot]

Modifie un état.

Parameters

<i>etat</i>	: l'état à modifier
<i>ini</i>	: true si l'état est initial
<i>final</i>	: true si l'état est final

#### 5.3.2.3 void CreateAutomate::displayRight ( int toDisplay ) [slot]

Affiche l'etat right spécifié

Parameters

<i>toDisplay</i>	: l'état à afficher
------------------	---------------------

#### 5.3.2.4 void CreateAutomate::supprimeEtat ( int cible ) [slot]

Supprime un état et réactualise l'affichage.

Parameters

<i>cible</i>	: l'état à supprimer
--------------	----------------------



### 5.3.3 Member Data Documentation

#### 5.3.3.1 Automate CreateAutomate::a

L'automate en train d'être construit

#### 5.3.3.2 vector<etatLeft\*> CreateAutomate::left

Vecteur représentant les états déjà construits

#### 5.3.3.3 QSvgWidget\* CreateAutomate::maVue

Partie inférieure droite de la fenetre, où l'automate est affiché

#### 5.3.3.4 vector<etatRight\*> CreateAutomate::right

Vecteur dont chaque élément représente une transition a modifié éventuellement pour un état

The documentation for this class was generated from the following files:

- /home/aaiighht/Bureau/projetTL/automate-project/createautomate.h
- /home/aaiighht/Bureau/projetTL/automate-project/createautomate.cpp

## 5.4 etat Class Reference

### Public Member Functions

- **etat** (int number, bool ini=false, bool fina=false)  
*Constructeur.*
- **etat** (const **etat** &e)  
*Constructeur.*
- **etat** ()  
*Destructeur.*
- int **getNumber** ()  
*Retourne le numéro de l'état.*
- void **setNumber** (int number)  
*Définit le numéro de l'état.*
- bool **isFinal** ()  
*Test si l'état est un état final.*
- void **setFinal** (bool decision)  
*Définit si l'état est final ou non.*
- bool **isInitial** ()  
*Test si l'état est un état initial.*
- void **setInitial** (bool decision)  
*Définit si l'état est initial ou non.*
- void **ajoutTransition** (**etat** cible, int vocab)  
*Ajoute une transition à l'état.*
- void **supprimeTransition** (**etat** cible, int vocab)  
*Supprime une transition à l'état.*
- void **renameTransition** (**etat** cible)  
*Renomme une transition de l'état.*

- `multimap< int, etat > getTransitions ()`  
*Retourne les transitions de l'état.*
- `bool operator== (etat &e) const`  
*Opérateur d'égalité entre 2 états.*
- `bool operator!= (etat &e) const`  
*Opérateur d'inégalité entre 2 états.*
- `bool find\_transition (int etiq, etat e)`  
*Test l'existence d'une transition.*
- `bool find\_transition (etat e)`  
*Test l'existence d'une transition.*
- `bool estDansList (list< etat > liste)`  
*Test l'état est dans une liste.*
- `void setName (string rename)`  
*Attribuer un nom à l'état.*
- `string getName ()`  
*Renvois le nom de l'état.*
- `string getNameF ()`  
*Renvois le nom de l'état.*
- `void setName (list< etat > l)`

## Public Attributes

- int [numero](#)

## 5.4.1 Constructor & Destructor Documentation

### 5.4.1.1 `etat::etat ( int number, bool ini = false, bool fin = false )`

Constructeur.

Construit l'état passé avec le numéro *number*, peut être initial et/ou final

Parameters

<i>number</i>	: numéro de l'état
<i>ini</i>	: par défaut false, true si l'état est initial
<i>fin</i>	: par défaut false, true si l'état est final

### 5.4.1.2 `etat::etat ( const etat & e )`

Constructeur.

Construit l'état passé en paramètre

Parameters

<i>e</i>	: pointeur vers l'état à construire
----------	-------------------------------------

### 5.4.1.3 `etat::etat ( ) [inline]`

Destructeur.

Destructeur de la classe Etat

## 5.4.2 Member Function Documentation

### 5.4.2.1 void etat::ajoutTransition ( etat *cible*, int *vocab* )

Ajoute une transition à l'état.

Ajoute une transition allant vers l'état cible et portant l'étiquette vocab.

Parameters

<i>cible</i>	: l'état vers laquelle la transition arrive
<i>vocab</i>	: l'étiquette de la transition

### 5.4.2.2 bool etat::estDansList ( list< etat > *liste* )

Test l'état est dans une liste.

Teste si l'état est dans la liste passée en paramètre. Se base sur l'opérateur d'égalité entre deux états.

Parameters

<i>liste</i>	: la liste d'états dans laquelle on va chercher l'état.
--------------	---

Returns

true si cette transition existe, false sinon

### 5.4.2.3 bool etat::find\_transition ( int *etiq*, etat *e* )

Test l'existence d'une transition.

Teste si la transition allant vers l'état e et portant l'étiquette etiq existe.

Parameters

<i>e</i>	: l'état d'arrivée de la transition
<i>etiq</i>	: l'étiquette de la transition

Returns

true si cette transition existe, false sinon

### 5.4.2.4 bool etat::find\_transition ( etat *e* )

Test l'existence d'une transition.

Teste si la transition allant vers l'état e existe.

Parameters

<i>e</i>	: l'état d'arrivée de la transition
----------	-------------------------------------

Returns

true si cette transition existe, false sinon

#### 5.4.2.5 `string etat::getName ( )`

Renvois le nom de l'état.

Renvoie le nom de l'état. Si aucun nom n'a été attribué, cela renvoie le numéro.

##### Returns

le nom de l'état

#### 5.4.2.6 `string etat::getNameF ( )`

Renvois le nom de l'état.

Fonctionne comme [getName\(\)](#) mais en ajoutant un F avant le nom de l'état.

##### Returns

le nom de l'état, précédé d'un F

#### 5.4.2.7 `int etat::getNumber ( )`

Retourne le numéro de l'état.

Getter de l'attribut number.

##### Returns

Le numéro de l'état

#### 5.4.2.8 `multimap< int, etat > etat::getTransitions ( )`

Retourne les transitions de l'état.

Getter de l'attribut transition. Le premier membre est l'étiquette de la transition et le deuxième est l'état ciblé par la transition.

##### Returns

Multimap représentant les transitions de l'état

#### 5.4.2.9 `bool etat::isFinal ( )`

Test si l'état est un état final.

Renvoie un booléen en fonction de si l'état est final ou non

##### Returns

true si l'état est final, false sinon

#### 5.4.2.10 `bool etat::isInitial ( )`

Test si l'état est un état initial.

Renvoie un booléen en fonction de si l'état est initial ou non

##### Returns

true si l'état est initial, false sinon

**5.4.2.11 bool etat::operator!= ( etat & e ) const**

Opérateur d'inégalité entre 2 états.

L'inégalité entre deux états se teste seulement sur le numéro de l'état.

**Parameters**

<i>e</i>	: l'état à tester
----------	-------------------

**Returns**

true si les numéros sont différents, false sinon

**5.4.2.12 bool etat::operator== ( etat & e ) const**

Opérateur d'égalité entre 2 états.

L'égalité entre deux états se teste seulement sur le numéro de l'état.

**Parameters**

<i>e</i>	: l'état à tester
----------	-------------------

**Returns**

true si les numéros sont les mêmes, false sinon

**5.4.2.13 void etat::renameTransition ( etat *cible* )**

Renomme une transition de l'état.

Renomme la transition allant vers l'état cible et portant l'étiquette vocab.

**Parameters**

<i>cible</i>	: l'état vers laquelle la transition arrive
<i>vocab</i>	: l'étiquette de la transition

**5.4.2.14 void etat::setFinal ( bool *decision* )**

Définit si l'état est final ou non.

Setter de l'attribut final.

**Parameters**

<i>decision</i>	: true si l'état est final, false sinon
-----------------	---

**5.4.2.15 void etat::setInitial ( bool *decision* )**

Définit si l'état est initial ou non.

Setter de l'attribut initial.

## Parameters

<i>decision</i>	: true si l'état est initial, false sinon
-----------------	---

5.4.2.16 void etat::setName ( string *rename* )

Attribuer un nom à l'état.

Setter sur l'attribut name.

## Parameters

<i>rename</i>	: le nom à donner à l'état
---------------	----------------------------

5.4.2.17 void etat::setNumber ( int *number* )

Définit le numéro de l'état.

Setter de l'attribut number.

## Parameters

<i>number</i>	: le numéro de l'état à mettre
---------------	--------------------------------

5.4.2.18 void etat::supprimeTransition ( etat *cible*, int *vocab* )

Supprime une transition à l'état.

Supprime la transition allant vers l'état cible et portant l'étiquette vocab.

## Parameters

<i>cible</i>	: l'état vers laquelle la transition arrive
<i>vocab</i>	: l'étiquette de la transition

## 5.4.3 Member Data Documentation

## 5.4.3.1 int etat::numero

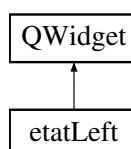
Numéro de l'état

The documentation for this class was generated from the following files:

- /home/aaiighht/Bureau/projetTL/automate-project/etat.h
- /home/aaiighht/Bureau/projetTL/automate-project/etat.cpp

## 5.5 etatLeft Class Reference

Inheritance diagram for etatLeft:



## Public Slots

- void [sendDad](#) ()  
*Listener du bouton permettant d'afficher l'etat right associé à cet état.*
- void [askForSupress](#) ()  
*Listener du bouton permettant la suppression de l'etat.*

## Signals

- void **selected** (int me)
- void **supress** (int)

## Public Member Functions

- [etatLeft](#) (int, QWidget \*parent=0)  
*Constructeur.*
- [~etatLeft](#) ()  
*Destructeur.*

## Protected Member Functions

- void **changeEvent** (QEvent \*e)

## Protected Attributes

- int [numero](#)

### 5.5.1 Constructor & Destructor Documentation

#### 5.5.1.1 `etatLeft::etatLeft ( int number, QWidget * parent = 0 )`

Constructeur.

Construit l'etat left

### 5.5.2 Member Data Documentation

#### 5.5.2.1 `int etatLeft::numero` [protected]

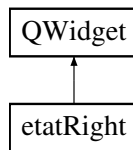
Numéro de l'état

The documentation for this class was generated from the following files:

- `/home/aaiighht/Bureau/projetTL/automate-project/etatleft.h`
- `/home/aaiighht/Bureau/projetTL/automate-project/etatleft.cpp`

## 5.6 etatRight Class Reference

Inheritance diagram for etatRight:



### Public Slots

- void [addTransition](#) (int to, int vocab)  
*Ajout d'une transition.*
- void [eraseTransition](#) (int to, int vocab)  
*Suppression d'une transition.*
- void **etatChange** ()

### Signals

- void **refreshNeeded** (int)
- void **etatChanges** (int, bool, bool)

### Public Member Functions

- [etatRight](#) ([Automate](#) \*a, int, QWidget \*parent=0)  
*Constructeur.*
- [~etatRight](#) ()  
*Destructeur.*
- void [remplirListChoix](#) ()  
*Remet à jour la liste dans [choixPointe](#) permettant de choisir l'etat cible de la transition.*
- void **addVisualTransition** (int, int)
- void **cleanTrans** ()

### Public Attributes

- [choixPointe](#) \* [addTrans](#)
- [Automate](#) \* a
- int [numero](#)

### Protected Member Functions

- void **changeEvent** (QEvent \*e)

#### 5.6.1 Constructor & Destructor Documentation

##### 5.6.1.1 etatRight::etatRight ( Automate \* a, int number, QWidget \* parent = 0 )

Constructeur.



## Parameters

<i>a</i>	: automate en cours de construction
----------	-------------------------------------

## 5.6.2 Member Function Documentation

5.6.2.1 void etatRight::addTransition ( int *to*, int *vocab* ) [slot]

Ajout d'une transition.

## Parameters

<i>to</i>	: numero de l'état cible
<i>vocab</i>	: etiquette de la transition

5.6.2.2 void etatRight::eraseTransition ( int *to*, int *vocab* ) [slot]

Suppresion d'une transition.

## Parameters

<i>to</i>	: numero de l'état cible
<i>vocab</i>	: etiquette de la transition

## 5.6.3 Member Data Documentation

## 5.6.3.1 Automate\* etatRight::a

Pointeur vers l'automate en construction

## 5.6.3.2 choixPointe\* etatRight::addTrans

Pointeur vers l'objet [choixPointe](#) permettant de gérer une transition

## 5.6.3.3 int etatRight::numero

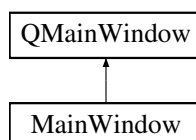
Numéro de l'état

The documentation for this class was generated from the following files:

- /home/aaiighht/Bureau/projetTL/automate-project/[etatright.h](#)
- /home/aaiighht/Bureau/projetTL/automate-project/etatright.cpp

## 5.7 MainWindow Class Reference

Inheritance diagram for MainWindow:



## Public Slots

- void [openFile](#) ()  
*Permet d'ouvrir un automate, stocké dans un fichier .dot.*
- void [creerAuto](#) ()  
*Permet d'accéder à la fenêtre de création d'automate.*
- void [getProduit](#) ()  
*Permet de faire le produit de 2 automates.*
- void [getDetermin](#) ()  
*Permet de déterminer un automate.*
- void [getSuivant](#) ()  
*Permet de passer à l'étape suivant pour une des opérations.*
- void [getPrecedent](#) ()  
*Permet de passer à l'étape précédente pour une des opérations.*
- void [getStandard](#) ()  
*Permet de standardiser un automate.*
- void [getMinimisation](#) ()  
*Permet de minimiser un automate.*
- void [resetUi](#) ()  
*Permet de nettoyer la fenetre.*
- void **test** ()
- void [info](#) ()  
*Permet d'afficher des informations sur les boutons et le fonctionnement du logiciel.*

## Public Member Functions

- [MainWindow](#) (QWidget \*parent=0)  
*Constructeur.*
- [~MainWindow](#) ()  
*Destructeur.*
- void **startLayouting** ()
- void **afficheAutomate** ([Automate](#))
- bool **lireDot** ()
- bool **lireDotB** ()

## Public Attributes

- QProcess \* **ProcessT**
- QString **program**

## Protected Member Functions

- void **changeEvent** (QEvent \*e)

### 5.7.1 Constructor & Destructor Documentation

#### 5.7.1.1 [MainWindow::MainWindow](#) ( [QWidget](#) \* *parent* = 0 )

Constructeur.

Construit la fenetre.

### 5.7.1.2 MainWindow::~~MainWindow ( )

Destructeur.

Destructeur de la classe [MainWindow](#)

## 5.7.2 Member Function Documentation

### 5.7.2.1 void MainWindow::creerAuto ( ) [slot]

Permet d'accéder à la fenêtre de création d'automate.

Listener du bouton ouvrir un automate, propose une fenetre pour créer un automate.

### 5.7.2.2 void MainWindow::getDetermin ( ) [slot]

Permet de déterminer un automate.

Listener du bouton déterminer. L'automate ouvert sera affiché dans maVue1. L'automate résultat du produit affiché dans maVue.

### 5.7.2.3 void MainWindow::getMinimisation ( ) [slot]

Permet de minimiser un automate.

Listener du bouton minimiser. L'automate ouvert sera affiché dans maVue1. L'automate résultat du produit affiché dans maVue.

### 5.7.2.4 void MainWindow::getPrecedent ( ) [slot]

Permet de passer à l'étape précédente pour une des opérations.

Listener du bouton précédent. L'étape précédente, s'il y en a une, d'une des opérations (standardisation, produit etc) sera affichée au lieu de l'actuel.

### 5.7.2.5 void MainWindow::getProduit ( ) [slot]

Permet de faire le produit de 2 automates.

Listener du bouton faire le produit de 2 automates, propose d'abord d'ouvrir une fenetre pour afficher le second automate. L'automate déjà ouvert sera affiché dans maVue1. L'automate ouvert sera affiché dans maVue2. L'automate résultat du produit sera affiché dans maVue.

### 5.7.2.6 void MainWindow::getStandard ( ) [slot]

Permet de standardiser un automate.

Listener du bouton standardiser. L'automate ouvert sera affiché dans maVue1. L'automate résultat du produit affiché dans maVue.

### 5.7.2.7 void MainWindow::getSuivant ( ) [slot]

Permet de passer à l'étape suivant pour une des opérations.

Listener du bouton suivant. L'étape suivante, s'il y en a une, d'une des opérations (standardisation, produit etc) sera affichée au lieu de l'actuel.

### 5.7.2.8 void MainWindow::info ( ) [slot]

Permet d'afficher des informations sur les boutons et le fonctionnement du logiciel.

Listener du bouton Info. Affiche dans la fenetre de droite des informations sur les boutons

### 5.7.2.9 void MainWindow::openFile ( ) [slot]

Permet d'ouvrir un automate, stocké dans un fichier .dot.

Listener du bouton ouvrir un automate, propose une fenetre pour sélectionner un fichier .dot correspondant à un automate. Cet automate sera ensuite afficher dans maVue

### 5.7.2.10 void MainWindow::resetUi ( ) [slot]

Permet de nettoyer la fenetre.

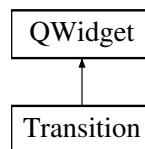
Listener du bouton nettoyer l'interface. Vide la fenetre et la ramène à un état comme au démarrage de l'application

The documentation for this class was generated from the following files:

- /home/aaiighht/Bureau/projetTL/automate-project/mainwindow.h
- /home/aaiighht/Bureau/projetTL/automate-project/mainwindow.cpp

## 5.8 Transition Class Reference

Inheritance diagram for Transition:



### Public Slots

- void [getOff](#) ()  
*Appelle la fonction eraser permettant de supprimer la transition.*

### Signals

- void **eraser** (int to, int [vocab](#))

### Public Member Functions

- [Transition](#) (int to, int [vocab](#), QWidget \*parent=0)  
*Constructeur.*
- [~Transition](#) ()  
*Destructeur.*

## Public Attributes

- int [cible](#)
- int [vocab](#)

## Protected Member Functions

- void **changeEvent** (QEvent \*e)

### 5.8.1 Constructor & Destructor Documentation

#### 5.8.1.1 Transition::Transition ( int *to*, int *vocab*, QWidget \* *parent* = 0 )

Constructeur.

Construit la transition passée en paramètre

Parameters

<i>to</i>	: numero de l'état cible de la transition
<i>vocab</i>	: etiquette de la transition

#### 5.8.1.2 Transition::~~Transition ( )

Destructeur.

Destructeur de la classe [Transition](#)

### 5.8.2 Member Data Documentation

#### 5.8.2.1 int Transition::cible

Numéro de l'état ciblé par la transition

#### 5.8.2.2 int Transition::vocab

Numéro de l'étiquette portée par la transition

The documentation for this class was generated from the following files:

- /home/aaiighht/Bureau/projetTL/automate-project/[transition.h](#)
- /home/aaiighht/Bureau/projetTL/automate-project/transition.cpp



## Chapter 6

# File Documentation

### 6.1 /home/aaiighht/Bureau/projetTL/automate-project/automate.h File Reference

Représente un automate, son seul attribut est un vector d'états.

```
#include <vector>
#include <set>
#include <list>
#include <map>
#include <string>
#include <cstdio>
#include <iostream>
#include <algorithm>
#include "etat.h"
```

#### Classes

- class [Automate](#)

#### Functions

- bool [equal](#) (list< [etat](#) > &l1, list< [etat](#) > &l2)  
*Test l'égalité entre deux listes d'états.*
- bool [isFinal](#) (list< [etat](#) > l)  
*Test si une liste d'états a au moins un état final.*

#### 6.1.1 Detailed Description

Représente un automate, son seul attribut est un vector d'états.

#### 6.1.2 Function Documentation

##### 6.1.2.1 bool [equal](#) ( list< [etat](#) > & l1, list< [etat](#) > & l2 )

Test l'égalité entre deux listes d'états.

Test l'égalité entre deux listes d'états l1 et l2

**Returns**

true si les listes sont égales, false sinon

**6.1.2.2 bool isFinal ( list< etat > / )**

Test si une liste d'états a au moins un état final.

Fonction utilisée seulement pour la détermination

**Parameters**

/	: liste d'états testée
---	------------------------

**Returns**

true s'il y a au moins un état final dans la liste d'états, false sinon

**6.2 /home/aaiighht/Bureau/projetTL/automate-project/choixpointe.h File Reference**

Utilisé pour la création d'un automate. Sert à définir les transitions en demandant à l'utilisateur de saisir vers quel état la transition va et quel étiquette porte cette transition.

```
#include <QWidget>
#include "automate.h"
```

**Classes**

- class [choixPointe](#)

**6.2.1 Detailed Description**

Utilisé pour la création d'un automate. Sert à définir les transitions en demandant à l'utilisateur de saisir vers quel état la transition va et quel étiquette porte cette transition.

**6.3 /home/aaiighht/Bureau/projetTL/automate-project/createautomate.h File Reference**

Représente la fenêtre principale lors de la création d'un automate, contient les autres éléments : etatleft, etatright, transition et choixpointe.

```
#include <QMainWindow>
#include "automate.h"
#include "etatleft.h"
#include "etatright.h"
#include <QSvgWidget>
#include <QTextBrowser>
#include <QFileDialog>
```

**Classes**

- class [CreateAutomate](#)



### 6.3.1 Detailed Description

Représente la fenêtre principale lors de la création d'un automate, contient les autres éléments : etatleft, etatright, transition et choixpointe.

## 6.4 /home/aaiighht/Bureau/projetTL/automate-project/etatleft.h File Reference

Représente la partie gauche lors de la création d'un automate, la partie listant les états.

```
#include <QWidget>
```

### Classes

- class [etatLeft](#)

### 6.4.1 Detailed Description

Représente la partie gauche lors de la création d'un automate, la partie listant les états.

## 6.5 /home/aaiighht/Bureau/projetTL/automate-project/etatright.h File Reference

Représente la partie sup droite lors de la création d'un automate, la partie listant les transitions d'un état, s'il est final/initial etc.

```
#include <QWidget>
#include "automate.h"
#include "choixpointe.h"
#include "transition.h"
```

### Classes

- class [etatRight](#)

### 6.5.1 Detailed Description

Représente la partie sup droite lors de la création d'un automate, la partie listant les transitions d'un état, s'il est final/initial etc.

## 6.6 /home/aaiighht/Bureau/projetTL/automate-project/mainwindow.h File Reference

Représente la fenetre principale du programme. On gère ici les listeners des boutons.

```
#include <QSvgWidget>
#include <QMainWindow>
#include <QProcess>
#include <vector>
#include "automate.h"
#include <QPushButton>
#include <QFileDialog>
#include <QTextBrowser>
#include <QScrollBar>
```

## Classes

- class [MainWindow](#)

### 6.6.1 Detailed Description

Représente la fenetre principale du programme. On gère ici les listeners des boutons.

## 6.7 /home/aaiiighht/Bureau/projetTL/automate-project/transition.h File Reference

Représente une transition d'un état.

```
#include <QWidget>
```

## Classes

- class [Transition](#)

### 6.7.1 Detailed Description

Représente une transition d'un état.

# Index

- ~Automate
  - Automate, [10](#)
- ~MainWindow
  - MainWindow, [28](#)
- ~Transition
  - Transition, [31](#)
- /home/aaighht/Bureau/projetTL/automate-project/automate.-
  - h, [33](#)
- /home/aaighht/Bureau/projetTL/automate-project/choixpointe.-
  - h, [34](#)
- /home/aaighht/Bureau/projetTL/automate-project/createautomate.-
  - h, [34](#)
- /home/aaighht/Bureau/projetTL/automate-project/etatleft.-
  - h, [35](#)
- /home/aaighht/Bureau/projetTL/automate-project/etatright.-
  - h, [35](#)
- /home/aaighht/Bureau/projetTL/automate-project/mainwindow.-
  - h, [35](#)
- /home/aaighht/Bureau/projetTL/automate-project/transition.-
  - h, [36](#)
- a
  - CreateAutomate, [19](#)
  - etatRight, [27](#)
- addTrans
  - etatRight, [27](#)
- addTransition
  - etatRight, [27](#)
- ajoutEtat
  - Automate, [10](#)
  - CreateAutomate, [18](#)
- ajoutTransition
  - Automate, [10](#)
  - etat, [21](#)
- Automate, [9](#)
  - ~Automate, [10](#)
  - ajoutEtat, [10](#)
  - ajoutTransition, [10](#)
  - Automate, [10](#)
  - cible\_transition, [12](#)
  - determinise, [12](#)
  - etats, [15](#)
  - getAlpha, [12](#)
  - getEtat, [12](#)
  - getEtats, [13](#)
  - getNbTransition, [13](#)
  - getTabTransitions, [13](#)
  - isDeterministe, [13](#)
  - isStandard, [13](#)
  - minimise, [13](#)
  - produit, [14](#)
  - standardise, [14](#)
  - supprimeEtat, [14](#)
  - supprimerEtatsNonAccessibles, [15](#)
  - toDot, [15](#)
- automate.h
  - equal, [33](#)
  - isFinal, [34](#)
- changeEvent
  - choixPointe, [16](#)
- changeState
  - CreateAutomate, [18](#)
  - choixPointe, [15](#)
- changeEvent, [16](#)
- choixPointe, [16](#)
- choixPointe, [16](#)
- resetAffichage, [16](#)
- sendDad, [16](#)
- cible
  - Transition, [31](#)
- cible\_transition
  - Automate, [12](#)
- CreateAutomate, [17](#)
  - a, [19](#)
  - ajoutEtat, [18](#)
  - changeState, [18](#)
  - CreateAutomate, [18](#)
  - CreateAutomate, [18](#)
  - displayRight, [18](#)
  - left, [19](#)
  - maVue, [19](#)
  - right, [19](#)
  - supprimeEtat, [18](#)
- creerAuto
  - MainWindow, [29](#)
- determinise
  - Automate, [12](#)
- displayRight
  - CreateAutomate, [18](#)
- equal
  - automate.h, [33](#)
- eraseTransition
  - etatRight, [27](#)
- estDansList
  - etat, [21](#)
- etat, [19](#)
  - ajoutTransition, [21](#)

- estDansList, 21
- etat, 20
- find\_transition, 21
- getName, 21
- getNameF, 22
- getNumber, 22
- getTransitions, 22
- isFinal, 22
- isInitial, 22
- numero, 24
- operator==, 23
- renameTransition, 23
- setFinal, 23
- setInitial, 23
- setName, 24
- setNumber, 24
- supprimeTransition, 24
- etatLeft, 24
  - etatLeft, 25
  - etatLeft, 25
  - numero, 25
- etatRight, 26
  - a, 27
  - addTrans, 27
  - addTransition, 27
  - eraseTransition, 27
  - etatRight, 26
  - etatRight, 26
  - numero, 27
- etats
  - Automate, 15
- find\_transition
  - etat, 21
- getAlpha
  - Automate, 12
- getDetermin
  - MainWindow, 29
- getEtat
  - Automate, 12
- getEtats
  - Automate, 13
- getMinimisation
  - MainWindow, 29
- getName
  - etat, 21
- getNameF
  - etat, 22
- getNbTransition
  - Automate, 13
- getNumber
  - etat, 22
- getPrecedent
  - MainWindow, 29
- getProduit
  - MainWindow, 29
- getStandard
  - MainWindow, 29
- getSuivant
  - MainWindow, 29
- getTabTransitions
  - Automate, 13
- getTransitions
  - etat, 22
- info
  - MainWindow, 29
- isDeterministe
  - Automate, 13
- isFinal
  - automate.h, 34
  - etat, 22
- isInitial
  - etat, 22
- isStandard
  - Automate, 13
- left
  - CreateAutomate, 19
- maVue
  - CreateAutomate, 19
- MainWindow, 27
  - ~MainWindow, 28
  - creerAuto, 29
  - getDetermin, 29
  - getMinimisation, 29
  - getPrecedent, 29
  - getProduit, 29
  - getStandard, 29
  - getSuivant, 29
  - info, 29
  - MainWindow, 28
  - MainWindow, 28
  - openFile, 30
  - resetUi, 30
- minimise
  - Automate, 13
- numero
  - etat, 24
  - etatLeft, 25
  - etatRight, 27
- openFile
  - MainWindow, 30
- operator==
  - etat, 23
- produit
  - Automate, 14
- renameTransition
  - etat, 23
- resetAffichage
  - choixPointe, 16
- resetUi
  - MainWindow, 30

- right
  - CreateAutomate, 19
- sendDad
  - choixPointe, 16
- setFinal
  - etat, 23
- setInitial
  - etat, 23
- setName
  - etat, 24
- setNumber
  - etat, 24
- standardise
  - Automate, 14
- supprimeEtat
  - Automate, 14
  - CreateAutomate, 18
- supprimeTransition
  - etat, 24
- supprimerEtatsNonAccessibles
  - Automate, 15
- toDot
  - Automate, 15
- Transition, 30
  - ~Transition, 31
  - cible, 31
  - Transition, 31
  - vocab, 31
- vocab
  - Transition, 31