Autoroute

Generated by Doxygen 1.8.6

Thu Feb 12 2015 16:56:24

Contents

Chapter 1

Documentation du logiciel Autoroute

1.1 Introduction

En allant dans la section Classes, vous aurait accès à la documentation de l'ensemble des classes. A partir de là, vous pourrez trouvez 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 vaux ne pas y toucher au début, surtout si l'on ne connait 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 Déterminisation de deux automates

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

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

Chapter 3

Class Index

3.1 Class List

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

Chapter 4

File Index

4.1 File List

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

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 chaine 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 déterminisation 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 ()

  Renvoit 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

a	: 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

cible	: 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

from	: l'état de départ de la transition
to	: l'état d'arrivée de la transition
vocab	: 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.

Renvoit 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

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

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 déterminisation de l'automate.

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

Returns

le vecteur servant pour la déterminisation

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 vector d'états (un attribut de l'automate).

Parameters

number	: 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 ( )

Renvoit 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 connaitre 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 renvoit un vecteur. Dans ce vecteur, chaque élément correspond à une pair : un automate et une chaine. Chaque élément pair représente en fait une étape dans le processus de minimisation. La chaine 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 renvoit un vecteur d'automates. Dans ce vecteur, chaque élément correspond à une étape du processus de produit de 2 automates.

Parameters

A	: 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 renvoit un vecteur. Dans ce vecteur, chaque élément correspond à une pair : un automate et une chaine. Chaque élément pair représente en fait une étape dans le processus de standardisation. La chaine 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

cible	: 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

cible	: l'état à supprimer
a	: 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

a	: 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 descrition DOT

Returns

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

5.1.3 Member Data Documentation

5.1.3.1 vector<etat> Automate::etats

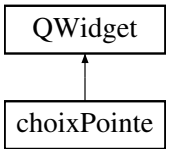
Vecteur des états de l'automate

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

- /home/aaiiighht/Bureau/projetTL/automate-project/automate.h
- /home/aaiiighht/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

e	: 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

a	: l'état qu'on est en train de construire

5.2.2.3 void choixPointe::sendDad ( ) [slot]

Listener du bouton +.

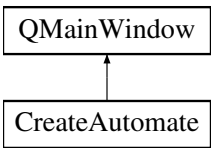
Méthode qui ajoute la transition seulement si les deux champs sont complétés. Listener du bouton +

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

- /home/aaiiighht/Bureau/projetTL/automate-project/choixpointe.h
- /home/aaiiighht/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

ini	: true si l'état est initial
fina	: true si l'état est final

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

Modifie un état.

Parameters

etat	: l'état à modifier
ini	: true si l'état est initial
final	: true si l'état est final

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

Affiche l'etat right spécifié

Parameters

toDisplay	: l'état à afficher

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

Supprime un état et réactualise l'affichage.

Parameters

cible	: 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/aaiiighht/Bureau/projetTL/automate-project/createautomate.h
- /home/aaiiighht/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 *fina* = `false`)

Constructeur.

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

Parameters

number	: numéro de l'état
ini	: par défaut false, true si l'état est initial
fina	: 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

e	: 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

cible	: l'état vers laquelle la transition arrive
vocab	: 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

liste	: 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

e	: l'état d'arrivée de la transition
etiq	: 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

e	: 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.

Renvoit le nom de l'état. Si aucun nom n'a été attribué, cela renvoit 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, précèdé 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 deuxieme 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.

Renvoit 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.

Renvoit 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

e	: 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

e	: 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

cible	: l'état vers laquelle la transition arrive
vocab	: 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

decision	: 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

decision	: 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

rename	: 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

number	: 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

cible	: l'état vers laquelle la transition arrive
vocab	: l'étiquette de la transition

5.4.3 Member Data Documentation

5.4.3.1 int etat::numero

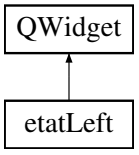
Numéro de l'état

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

- /home/aaiiighht/Bureau/projetTL/automate-project/etat.h
- /home/aaiiighht/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]

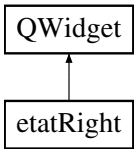
Numéro de l'état

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

- /home/aaiiighht/Bureau/projetTL/automate-project/etatleft.h
- /home/aaiiighht/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)

  Suppresion 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

a	: 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

to	: numero de l'état cible
vocab	: etiquette de la transition

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

Suppresion d'une transition.

Parameters

to	: numero de l'état cible
vocab	: 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

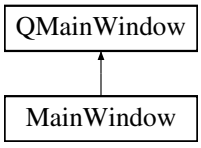
Numéro de l'état

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

- /home/aaiiighht/Bureau/projetTL/automate-project/etatright.h
- /home/aaiiighht/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 déterminiser 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 déterminiser un automate.

Listener du bouton déterminiser. 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.

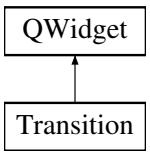
Listener du bouton nettoyer l'interface. Vide la fenetre et la ramène à un état comme au démarrage de l'application

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

- /home/aaiiighht/Bureau/projetTL/automate-project/mainwindow.h
- /home/aaiiighht/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

to	: numero de l'état cible de la transition
vocab	: 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/aaiiighht/Bureau/projetTL/automate-project/transition.h
- /home/aaiiighht/Bureau/projetTL/automate-project/transition.cpp

Chapter 6

File Documentation

6.1 /home/aaiiighht/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 > *l*)

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

Fonction utilisée seulement pour la déterminisation

Parameters

l	: 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/aaiiighht/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/aaiiighht/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/aaiiighht/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/aaiiighht/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/aaiiighht/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