Republic of
Yemen
IBB University
Faculty of Science
Department of IT
Theory of
Computation

الجمهورية اليمنية
جامعة إب
كلية العلوم التطبيقية
قسم : تقنية معلومات
المادة : نظرية الحوسبة

تقرير مشروع النظرية الإحتسابية

إشراف د : خالد الكحسة

الطلاب /

أيمن محمد قمحان – طارق فضل العمري

حازم هزام العمري – ضياء فضل الحضرمي – علي محمد القواس

عبدالله عبدالملك فارع

٢٠٢٥ م

عنوان المشروع:

تحليل وتصميم وتنفيذ محاكي النماذج الحوسبية

(Finite Automata Pushdown Automata Turing Machine)

باستخدام لغة C# ومنصة Windows Forms.

---

٢. المقدمة

٢,١ نظرة عامة على المشروع وأهدافه التعليمية

يُمثل هذا المشروع نظام برمجي تعليمي يهدف إلى محاكاة المفاهيم الأساسية في نظرية الحوسبة والأوتوماتا. يُعرف هذا المجال بكونه حجر الأساس في علوم الحاسب، حيث يقدم النماذج المجردة للآلات الحوسبية وقدراتها على حل المشكلات. ومع ذلك، فإن الطبيعة المجردة لهذه النماذج تجعل فهمها تحديًا للطلاب عند الاعتماد على المصادر النظرية فقط.

يهدف هذا النظام إلى تجسير الفجوة بين النظرية والتطبيق من خلال توفير بيئة تفاعلية ومرئية. يتيح النظام للمستخدمين (الطلاب والمهتمين) بناء واختبار ثلاثة من أهم النماذج الحوسبية:

١. **الآلات المنتهية (Finite Automata - FA):** وهي أبسط النماذج، وتُستخدم للتعرف على اللغات النمطية (Regular Languages) وتعتبر أساسًا في تصميم المحللات اللغوية (Lexical Analyzers).

٢. **آلات الدفع للأسفل (Pushdown Automata - PDA):** وهي امتداد للآلات المنتهية مع إضافة ذاكرة "مكدس" (Stack)، مما يمكنها من التعرف على اللغات حرة السياق (Context-Free Languages).

٣. **آلة تورنغ (Turing Machine - TM):** وهي النموذج الأكثر قوة وقدرة، حيث تمثل التعريف الرياضي للحاسوب متعدد الأغراض. يمكنها محاكاة أي خوارزمية حاسوبية.

الهدف الأساسي للمشروع هو تعزيز الفهم العملي لهذه النماذج من خلال تمكين المستخدم من:

- تعريف الآلات بطرق مختلفة (كتعبير نمطي للـ FA، أو قواعد انتقالات للـ PDA و TM).
- مشاهدة تمثيل مرئي (رسم بياني) للآلة التي تم بناؤها.
- الاطلاع على جدول الانتقالات الرسمي للآلة.
- اختبار سلوك الآلة عبر إدخال سلاسل نصية مختلفة ومراقبة النتيجة (قبول أو رفض).
- تتبع خطوات المحاكاة (خاصة في PDA و TM) لفهم عملية المعالجة الداخلية.

٢,٢ أهمية محاكاة النماذج الحوسبية

تكمن أهمية هذا المحاكي في قدرته على تحويل المفاهيم الرياضية المجردة إلى كائنات برمجية ملموسة وتفاعلية. فبدلاً من تتبع الحالات والانتقالات على الورق، يمكن للطالب أن يرى الآلة وهي "تعمل" أمامه، مما يوفر الفوائد التالية:

- **تعميق الفهم:** تساعد الرؤية المرئية لعملية الانتقال بين الحالات، أو التلاعب بالمكدس في PDA، أو حركة رأس القراءة/الكتابة على الشريط في TM، على ترسيخ المفاهيم بشكل أعمق.
- **التجربة والاكتشاف:** يمنح النظام المستخدم حرية تجربة تعابير وسلاسل معقدة ورؤية نتائجها فورًا، مما يشجع على التعلم القائم على الاستكشاف.
- **تصحيح الأخطاء المفاهيمية:** عندما يفشل اختبار سلسلة معينة، يمكن للمستخدم مراجعة خطوات المحاكاة (Trace) لفهم أين حدث الخطأ في منطقه أو في تصميم الآلة.

- **أداة مساعدة للتدريس:** يمكن للمحاضرين استخدام المحاكي كأداة عرض فعالة في الفصول الدراسية لشرح سلوك الآلات المعقدة بطريقة ديناميكية.

---

٣. دراسة الجدوى الأولية

٣,١ وصف المشكلة في النظام التقليدي (قبل الأتمتة)

يعتمد النظام التعليمي التقليدي في تدريس نظرية الحوسبة على الوسائل التالية:

- **الكتب والمحاضرات النظرية:** تقدم هذه الوسائل الأساس الرياضي ولكنها تفتقر إلى التفاعل.
- **الحل اليدوي على الورق:** يقوم الطلاب برسم الآلات وتتبع السلاسل يدويًا. هذه العملية بطيئة، وعرضة للأخطاء، وتصبح شبه مستحيلة مع الآلات المعقدة أو السلاسل الطويلة.
- **غياب التغذية الراجعة الفورية:** عند حل مسألة يدويًا، لا يعرف الطالب ما إذا كان حله صحيحًا أم لا حتى يتم تصحيحه من قبل المحاضر.

هذه الطرق تخلق صعوبات جمة لدى الطلاب، حيث يجدون أنفسهم غارقين في تفاصيل رياضية معقدة دون القدرة على بناء تصور عملي لكيفية عمل هذه الآلات، مما يؤدي إلى فهم سطحي للمادة.

٣,٢ الفوائد المتوقعة من النظام المحاكي

يهدف النظام المقترح (المحاكي) إلى التغلب على قيود النظام التقليدي من خلال تقديم الفوائد التالية:

- **الفعالية التعليمية:** تحويل عملية التعلم من عملية سلبية (تلقي المعلومة) إلى عملية إيجابية (تفاعلية)، مما يرفع من مستوى استيعاب الطلاب.
- **توفير الوقت والجهد:** أتمتة عملية بناء الآلات واختبارها توفر وقتًا كبيرًا كان يُهدر في الرسم والتتبع اليدوي.
- **الدقة والموثوقية:** يقدم النظام نتائج دقيقة وموثوقة، ويزيل احتمالية الخطأ البشري في تتبع العمليات المعقدة.
- **التغذية الراجعة الفورية:** يحصل المستخدم على نتيجة (مقبولة/مرفوضة) فورًا بعد اختبار أي سلسلة، مع سجل تتبع يوضح "لماذا" تم التوصل إلى هذه النتيجة.
- **المرئيات التوضيحية:** يقدم النظام تمثيلاً بيانياً (State Diagram) وجداول انتقالات يتم إنشاؤها تلقائيًا، مما يساعد على فهم بنية الآلة بشكل أفضل.

---

٤. تحليل النظام (System Analysis)

٤,١ متطلبات النظام (System Requirements)

أ) المتطلبات الوظيفية (Functional Requirements):

- **FR1:** يجب أن يسمح النظام للمستخدم باختيار نوع الآلة المراد محاكاتها (FA PDA TM).
- **FR2:** يجب أن يسمح النظام للمستخدم باختيار النوع الفرعي للآلة (Deterministic/Non-deterministic) بالنسبة لـ FA و PDA.
- **FR3 (FA):** يجب أن يتمكن النظام من بناء آلة منتهية (NFA أو DFA) من تعبير نمطي (Regular Expression) يُدخله المستخدم.
- **FR4 (PDA):** يجب أن يتمكن النظام من بناء آلة دفع للأسفل من مجموعة قواعد انتقالات نصية يدخلها المستخدم.
- **FR5 (TM):** يجب أن يتمكن النظام من بناء آلة تورنغ من مجموعة قواعد انتقالات نصية يدخلها المستخدم.

- **FR6:** يجب أن يعرض النظام تمثيلاً بيانياً (State Diagram) للآلة التي تم بناؤها.
- **FR7:** يجب أن يعرض النظام جدول انتقالات (Transition Table) يصف سلوك الآلة.
- **FR8:** يجب أن يسمح النظام للمستخدم بإدخال سلسلة نصية لاختبارها على الآلة.
- **FR9:** يجب أن يعرض النظام نتيجة الاختبار بوضوح (مقبولة / Accepted أو مرفوضة / Rejected).
- **FR10 (PDA):** يجب أن يعرض النظام سجل تتبع (Trace Log) يوضح خطوات المحاكاة لتشغيل سلسلة الإدخال.
- **FR11 (TM):** يجب أن يعرض النظام تمثيلاً مرئياً للشريط (Tape) وحركة رأس القراءة/الكتابة أثناء المحاكاة.
- **FR12:** يجب أن يتحقق النظام من صحة صيغة الإدخال (للتعابير النمطية وقواعد الانتقالات) وإعلام المستخدم بالأخطاء.

ب) المتطلبات غير الوظيفية (Non-Functional Requirements):

- **NFR1 (Usability):** يجب أن تكون واجهة المستخدم سهلة الفهم والاستخدام لطلاب علوم الحاسب.
- **NFR2 (Performance):** يجب أن يقوم النظام ببناء الآلات وتشغيل المحاكاة في زمن استجابة معقول (أقل من بضع ثوانٍ للحالات المتوسطة).
- **NFR3 (Reliability):** يجب أن يكون النظام مستقرًا ولا ينهار عند إدخال مدخلات غير متوقعة (مع معالجة الأخطاء بشكل سليم).
- **NFR4 (Clarity):** يجب أن تكون المخرجات المرئية (الرسوم البيانية، الجداول، النتائج) واضحة وسهلة القراءة.
- **NFR5 (Platform):** يجب أن يعمل النظام على أنظمة تشغيل ويندوز التي تدعم بيئة .NET Framework.

٤,٢ تحديد الجهات الفاعلة (Actors)

يوجد في النظام جهة فاعلة رئيسية واحدة:

- **المستخدم (User):** يمكن أن يكون طالبًا في مساق نظرية الحوسبة، أو مدرسًا يستخدم الأداة للشرح، أو أي شخص مهتم باستكشاف هذه النماذج. يتفاعل المستخدم مع النظام لتحديد الآلات، واختبارها، وفهم سلوكها.

٤,٣ مخطط حالة الاستخدام (Use Case Diagram)

لا يمكن رسم المخطط هنا، ولكن يمكن وصفه نصيًا. الجهة الفاعلة هي "المستخدم". حالات الاستخدام الرئيسية هي:

- **Select Machine Type**: يختار المستخدم بين (FA PDA TM).
- **Define Finite Automaton**: يدخل المستخدم تعبيرًا نمطيًا. وهذه الحالة تتضمن (BuildFaButton_Click).
- **Define Pushdown Automaton**: يدخل المستخدم قواعد انتقالات PDA. وهذه الحالة تتضمن (BuildPdaButton_Click).
- **Define Turing Machine**: يدخل المستخدم قواعد انتقالات TM. وهذه الحالة تتضمن (BuildTmButton_Click).
- **Test Input String**: يدخل المستخدم سلسلة نصية ويطلب اختبارها. (مرتبطة بالثلاث حالات السابقة).
- **View State Diagram**: يقوم النظام بعرض الرسم البياني للآلة. (نتيجة لحالات التعريف).
- **View Transition Table**: يقوم النظام بعرض جدول الانتقالات. (نتيجة لحالات التعريف).

- **View Simulation Trace**: يقوم النظام بعرض خطوات المحاكاة. (نتيجة لاختبار السلسلة في PDA و TM).

٤,٤ مخططات تدفق البيانات (Data Flow Diagrams - DFD)

المستوى صفر (Context Diagram):

- **كيان خارجي (External Entity):** User.
- **عملية (Process):** ٠. Computational Theory Simulator.
- **تدفقات بيانات واردة (Incoming Data Flows):** Machine_Definition (Regex PDA Rules TM Rules) Test_String.
- **تدفقات بيانات صادرة (Outgoing Data Flows):** Transition_Table Visual_Diagram Simulation_Trace Test_Result.

المستوى الأول (Level 1 DFD):

تتفرع العملية ٠ إلى ثلاث عمليات رئيسية:

- **Process ١,٠ Manage FA:** يستقبل Regex من المستخدم، ويُخرج FA_Diagram و FA_Table.
- **Process ٢,٠ Manage PDA:** يستقبل PDA_Rules من المستخدم، ويُخرج PDA_Diagram و PDA_Table.
- **Process ٣,٠ Manage TM:** يستقبل TM_Rules من المستخدم، ويُخرج TM_Diagram و TM_Table.
- **Process ٤,٠ Simulate Machine:** يستقبل Test_String من المستخدم، ويستقبل Machine_Model من العمليات (١,٠ ٢,٠ ٣,٠)، ويُخرج Test_Result و Simulation_Trace.
- **مخازن بيانات (Data Stores):** يمكن تصور وجود مخازن بيانات مؤقتة مثل :D1 Current_FA_Model D2: Current_PDA_Model D3: Current_TM_Model.

٤,٥ تحديد الكيانات والعمليات والمخرجات الرئيسية

- **الكيانات (Entities/Models):** State PushdownAutomaton TuringMachine PDATransition TMTransition.
- **العمليات الرئيسية (Processes):**
  - التحليل (Parsing): ParsePdaDefinition ParseTm.
  - التحويل (Conversion): InfixToPostfix PostfixToNfa NfaToDfa.
  - المحاكاة (Simulation): SimulatePda RunTmSimulation TestDfaString.
  - العرض المرئي (Visualization): FaDiagramPanel_Paint PdaDiagramPanel_Paint TmTapeVisualizer_Paint.
- **المخرجات الرئيسية (Outputs):**
  - الرسوم البيانية للحالات (State Diagrams).
  - جداول الانتقالات (Transition Tables).

- نتائج اختبار السلاسل (Accepted/Rejected).
- سجلات التتبع المرئية (Visual Traces and Tape).

---

٥. تصميم النظام (System Design)

٥,١ التصميم المنطقي للنظام (Conceptual Data Model)

على الرغم من أن النظام لا يستخدم قاعدة بيانات تقليدية، يمكننا تصميم نموذج بيانات مفاهيمي يوضح العلاقات بين الكائنات الرئيسية في الذاكرة.

- **كيان Machine (مفهوم عام):**
  - يمكن أن يكون FiniteAutomaton PushdownAutomaton أو TuringMachine.
  - يحتوي على مجموعة من States وحالة بداية StartState.
- **كيان State:**
  - له Id (معرف).
  - له خاصية IsAcceptState (بوليانية).
  - له مجموعة من Transitions. (علاقة واحد إلى متعدد: State -> Transition).
- **كيان Transition (مفهوم عام):**
  - يربط بين FromState و ToState.
  - **في FA:** يعتمد على InputSymbol.
  - **في PDA:** يعتمد على InputSymbol StackPopSymbol وينتج StackPushSymbols.
  - **في TM:** يعتمد على ReadSymbol وينتج WriteSymbol و MoveDirection.

هذا التصميم المنطقي ينعكس مباشرة في الكود من خلال الفئات State PDATransition TMTransition وكيفية تجميعها داخل PushdownAutomaton و TuringMachine.

٥,٢ تصميم الواجهة الرسومية (GUI Design)

تم تصميم الواجهة لتكون مقسمة ومنظمة.

- **القسم العلوي:** يحتوي على أزرار RadioButton لاختيار نوع الآلة (rbFiniteAutomata rbPushdownAutomata rbTuringMachine) والنوع الفرعي (rbDeterministic rbNondeterministic). هذا الاختيار يحدد ديناميكيًا محتوى اللوحة الرئيسية.
- **اللوحة الرئيسية (mainContentPanel):** تتغير محتوياتها بناءً على الاختيار. بشكل عام، يتم تقسيمها إلى قسمين:
  - **القسم الأيسر:** مخصص لتعريف الآلة (صندوق نص لإدخال القواعد أو التعبير النمطي) وأسفله لوحة لعرض الرسم البياني (faDiagramPanel pdaDiagramPanel tmDiagramPanel).
  - **القسم الأيمن:** مخصص للنتائج والتفاعل. يحتوي على جدول الانتقالات (DataGridView)، ومنطقة لاختبار السلاسل، وعرض النتيجة النهائية، وسجل التتبع أو شريط تورنغ.

- **استخدام TableLayoutPanel و SplitContainer:** يوفر مرونة في تقسيم المساحة ويضمن أن الواجهة تتكيف بشكل جيد مع تغيير حجم النافذة.

٥,٣ تفاعل المستخدم مع الواجهات

١. **الاختيار:** يبدأ المستخدم باختيار نوع الآلة (مثل PDA).

٢. **التعريف:** يكتب المستخدم قواعد الانتقالات في pdaTransitionsInput. على سبيل المثال، q0 a Z q1 AZ لتعريف انتقال من q0 إلى q1 عند قراءة a وإزالة Z من المكدس، ثم إضافة AZ.

٣. **البناء:** يضغط المستخدم على زر "بناء الآلة". يقوم النظام بتشغيل ParsePdaDefinition و BuildPda، مما يؤدي إلى تحديث الرسم البياني (pdaDiagramPanel.Invalidate()) وجدول الانتقالات (UpdatePdaTransitionTable).

٤. **الاختبار:** يكتب المستخدم سلسلة في pdaTestStringInput (مثل "aaabbb").

٥. **التشغيل:** يضغط على "اختبر". يقوم النظام بتشغيل SimulatePda.

٦. **النتائج:** يتم تحديث pdaResultLabel بالنتيجة، ويتم ملء pdaTraceLog بخطوات المحاكاة التفصيلية.

٥,٤ آلية معالجة المدخلات وتحليلها

- **FA Input (Regex):** تستخدم الدالة AddConcatOperator لإضافة عامل الربط . بشكل صريح، ثم تحول الدالة InfixToPostfix التعبير إلى صيغة لاحقة (postfix) باستخدام خوارزمية Shunting-yard. هذه الصيغة أسهل للمعالجة.
- **PDA/TM Input (Rules):** تستخدم الدالة ParsePdaDefinition و ParseTm تعابير نمطية (Regex) لتحليل كل سطر من تعريف المستخدم. على سبيل المثال، النمط @" *s(.)s* s*(.)s* s*q(d+)s* s *s*q(d+)s*(.+)" في PDA يمسك بكل جزء من القاعدة (الحالة الحالية، رمز الإدخال، رمز الإزالة، الحالة التالية، رموز الإضافة) ويحولها إلى كائن PDATransition.

٥,٥ نموذج سلوك الآلات (Behavior Model)

- **Finite Automata:**
  - **DFA:** يتم تتبعه بشكل مباشر. لكل رمز في سلسلة الإدخال، ينتقل من الحالة الحالية إلى حالة واحدة تالية محددة.
  - **NFA:** يستخدم النظام مجموعة من الحالات النشطة (currentStates). مع كل رمز إدخال، يحدد النظام كل الحالات الممكنة التي يمكن الوصول إليها من المجموعة الحالية (Move)، ثم يوسع هذه المجموعة لتشمل كل الحالات التي يمكن الوصول إليها عبر انتقالات إبسيلون (EpsilonClosure).
- **Pushdown Automaton:**
  - يتم استخدام طابور (Queue) من الكيانات PDAConfiguration لمحاكاة السلوك غير المحدد. كل PDAConfiguration يمثل "فرعًا" محتملاً من الحوسبة (بحالته الحالية، ومؤشر الإدخال، ومحتوى المكدس).
  - في كل خطوة، يأخذ النظام تكوينًا من الطابور، ويجد كل الانتقالات الممكنة، وينشئ تكوينات جديدة لكل انتقال ويضيفها إلى الطابور. تستمر العملية حتى يتم قبول السلسلة أو استنفاد جميع المسارات الممكنة.
- **Turing Machine:**
  - يتم تمثيل الشريط كـ Dictionary<int char>، حيث يمثل المفتاح موقع الخلية.

- تحاكي الحلقة RunTmSimulation دورة عمل الآلة: قراءة الرمز تحت الرأس، البحث عن الانتقال المناسب في Transitions كتابة الرمز الجديد، تحديث الحالة الحالية، وتحريك الرأس يسارًا أو يمينًا. تستمر الدورة حتى الوصول إلى حالة القبول أو الرفض.

. شرح الكود البرمجي

٦,١ البنية المعمارية للمشروع (Architecture)

يتبع المشروع بنية منطقية يمكن تقسيمها إلى ثلاث طبقات داخل التطبيق نفسه:

١. **طبقة العرض (Presentation Layer):**

- **المسؤولية:** عرض واجهة المستخدم، والتقاط تفاعلات المستخدم.
- **المكونات:** الفئة testMain وكل عناصر التحكم (Control) الموجودة فيها مثل الأزرار ومربعات النصوص. الدوال مثل SetupUI CreateFiniteAutomataPanel وكل معالجات الأحداث *_Click *_Paint هي جزء من هذه الطبقة.

٢. **طبقة منطق الأعمال (Business Logic Layer):**

- **المسؤولية:** تنفيذ القواعد والعمليات الأساسية للنظام.
- **المكونات:** الدوال التي تحتوي على الخوارزميات الرئيسية مثل NfaToDfa InfixToPostfix RunTmSimulation ParsePdaDefinition SimulatePda. هذه الطبقة هي قلب النظام.

٣. **طبقة البيانات/النماذج (Data/Model Layer):**

- **المسؤولية:** تمثيل البيانات والكيانات التي يعمل عليها النظام.
- **المكونات:** الفئات البسيطة (POCOs) التي تصف بنية الآلات: NfaFragment State TMTransition TuringMachine PDATransition PushdownAutomaton والفئة المساعدة VisualState.

هذا الفصل (ولو كان منطقيًا فقط) يجعل الكود أكثر تنظيمًا وقابلية للصيانة.

٦,٢ شرح أهم الكلاسات والوظائف

- **State:**
  - يمثل حالة في آلة منتهية.
  - يحتوي على Id رقمي، IsAcceptState لتحديد ما إذا كانت حالة نهائية.
  - الأهم هو Transitions وهو Dictionary<State>char List>> الذي يربط رمز الإدخال بقائمة من الحالات التالية (قائمة لدعم السلوك غير المحدد).
- **PushdownAutomaton:**
  - يمثل آلة الدفع للأسفل بأكملها.

- يحتوي على Transitions (قاموس يربط رقم الحالة بقائمة من انتقالاتها PDATransition)، StartStateId، ومجموعة من AcceptStates.

- **TuringMachine:**
  - يمثل آلة تورنغ.
  - يحتوي على Transitions وهو قاموس مفتاحه TMTransitionKey (يجمع بين الحالة والرمز المقروء) وقيمته TMTransition (يحتوي على الحالة التالية، والرمز المكتوب، واتجاه الحركة).
  - يحتوي أيضًا على StartStateId AcceptStateId RejectStateId.

- **SimulatePda(..):**
  - وظيفة حيوية تنفذ محاكاة PDA. تستخدم خوارزمية البحث بالعرض (Breadth-First Search) عن طريق Queue لاستكشاف جميع مسارات الحوسبة الممكنة في حالة الآلات غير المحددة، مما يضمن العثور على مسار مقبول إن وجد.

- **ParsePdaDefinition(..):**
  - توضح قوة التعابير النمطية (Regex) في تحليل النصوص المهيكلة. تقوم بتحليل تعريفات المستخدم، والتحقق من صحتها، وتحويلها إلى كائنات PDATransition التي يفهمها النظام.

٦,٣ تحليل بناء الرسوم البيانية، آلية الاختبار، والمحاكاة

- **بناء الرسوم البيانية:**

١. **تحديد الحالات:** بعد بناء الآلة (مثلاً currentPda)، يتم جمع كل أرقام الحالات الفريدة من جداول الانتقالات.

٢. **إنشاء كائنات مرئية:** يتم إنشاء قائمة من List<VisualState>، حيث يمثل كل VisualState حالة على الرسم.

٣. **تحديد المواقع:** تستدعي الدالة PositionStates أو PositionPdaStates التي تقوم بتوزيع الحالات بشكل دائري على اللوحة (Panel) لتجنب التداخل.

٤. **الرسم:** عندما يحتاج Panel إلى إعادة رسم نفسه (بعد استدعاء Invalidate())، يتم تشغيل معالج الحدث Paint_*.

- **آلية الرسم (GDI+):**

١. يتم الحصول على كائن Graphics من e.Graphics.

٢. يتم رسم الانتقالات أولاً كخطوط (DrawLine) أو أقواس (DrawArc للحلقات الذاتية). يتم حساب نقطة المنتصف لوضع التسمية (label).

٣. يتم رسم الحالات ثانيًا كدوائر (DrawEllipse FillEllipse) فوق الخطوط لتبدو متصلة بشكل صحيح.

٤. تُستخدم أقلام (Pen) وفرش (Brush) وخطوط (Font) مختلفة للتمييز بين الحالات العادية والنهائية والانتقالات.

٦,٤ شرح طريقة رسم الحالات والانتقالات (GDI+)

توضح الدالة DrawPdaDiagram هذه العملية بالتفصيل:

١. **إعدادات الجودة:** e.Graphics.SmoothingMode = SmoothingMode.AntiAlias لتحسين مظهر الخطوط والمنحنيات.

٢. **تجميع الانتقالات:** يتم تجميع الانتقالات حسب الحالة المصدر والهدف لتجنب رسم خطوط متعددة بين نفس الحالتين.

٣. **رسم الانتقالات:**

- DrawSelfLoop: ترسم حلقة ذاتية باستخدام DrawArc.
- DrawTransitionBetweenStates: ترسم خطًا مباشرًا باستخدام DrawLine وتضيف رأس سهم باستخدام DrawArrowHead.
- FormatTransitionLabel: تقوم بإنشاء نص التسمية للانتقال بالصيغة input pop/push.

٤. **رسم الحالات:**

- DrawState: ترسم دائرة أساسية، ودائرة خارجية إضافية إذا كانت حالة قبول (pda.AcceptStates.Contains(state.Id)).
- DrawStartArrow: ترسم سهمًا يشير إلى حالة البداية.

---

٧. التقييم والتحسين

٧,١ نقاط القوة الحالية

- **الشمولية:** يدعم النظام ثلاثة أنواع رئيسية من الآلات، مما يجعله أداة شاملة.
- **التفاعلية والمرئية:** يوفر النظام تغذية راجعة مرئية فورية، وهي أفضل نقاط قوته التعليمية.
- **الفصل بين المنطق والعرض:** استخدام فئة VisualState منفصلة عن نماذج الآلات هو ممارسة تصميم جيدة.
- **معالجة عدم التحديدية (Non-determinism):** آلية المحاكاة في PDA قوية وقادرة على التعامل مع السلوك غير المحدد بشكل صحيح.
- **سهولة الإدخال:** استخدام التعابير النمطية لـ FA وقواعد نصية بسيطة لـ PDA و TM يجعل تعريف الآلات سريعًا.

٧,٢ تحديات أو قيود

- **محدودية الإدخال:** لا يمكن للمستخدم بناء آلة (خاصة FA) عن طريق رسمها مباشرة أو تحديد حالاتها وانتقالاتها يدويًا؛ هو مقيد بالتعابير النمطية فقط.
- **تخطيط الرسم البياني:** التخطيط الدائري التلقائي قد لا يكون مثاليًا للرسوم البيانية المعقدة، وقد تبدو بعض الانتقالات متقاطعة أو مزدحمة.
- **غياب الحفظ والتحميل:** لا يمكن للمستخدم حفظ تعريف آلة معقدة والعودة إليها لاحقًا، مما يتطلب إعادة إدخالها في كل مرة.
- **المحاكاة خطوة بخطوة:** المحاكاة تعمل بشكل كامل (RunTmSimulation) أو تعرض سجلًا بعد الانتهاء (pdaTraceLog). سيكون من الأفضل وجود تحكم خطوة بخطوة (Step-by-step) يسمح للمستخدم بتنفيذ انتقال واحد في كل مرة.

٧,٣ مقترحات تطويرية مستقبلية

من منظور تحليل وتصميم النظم، يمكن تحسين النظام بإضافة الميزات التالية:

- **محرر رسومي للآلات:** إضافة وظيفة تسمح للمستخدم بسحب وإفلات الحالات ورسم الانتقالات بينها باستخدام الفأرة. هذا سيضيف متطلبًا وظيفيًا جديدًا كبيرًا ويتطلب تصميمًا دقيقًا لواجهة المستخدم ومنطق العمل.
- **وظيفة الحفظ والتحميل:** إضافة خيار لحفظ تعريف الآلة الحالي في ملف (مثل XML أو JSON) واسترجاعه لاحقًا. هذا يتطلب تصميم مخطط (Schema) للملفات وإضافة وحدات للـ Serialization/Deserialization.

- **تحسين المحاكاة:**
  - إضافة أزرار (Play Pause Step Forward) للتحكم في المحاكاة.
  - تمييز الحالة النشطة والانتقال المستخدم في الرسم البياني بشكل مرئي أثناء المحاكاة خطوة بخطوة.
- **دعم أنواع إضافية:** توسيع النظام ليشمل آلات أخرى مثل آلات ميلي (Mealy) ومور (Moore) أو آلات تورنغ متعددة الأشرطة.
- **مكتبة أمثلة:** تضمين مجموعة من الأمثلة الجاهزة للغات معروفة (مثل anbn) يمكن للمستخدم تحميلها لدراستها.

---

٨. الخاتمة

٨,١ ملخص لما تم إنجازه

نجح هذا المشروع في تطوير نظام محاكاة فعال وشامل لنماذج الحوسبة الأساسية. تم تحليل متطلبات النظام وتصميم بنية برمجية قوية ومنظمة باستخدام لغة C#، مع واجهة مستخدم رسومية تفاعلية مبنية على Windows Forms. يقوم النظام بترجمة التعريفات النظرية للآلات إلى تمثيلات مرئية وجداول دقيقة، ويسمح باختبارها وتتبع سلوكها، مما يحقق الأهداف التعليمية المرجوة بالكامل.

٨,٢ الفائدة العلمية للمشروع

يقدم المشروع مساهمة قيمة في مجال التعليم الهندسي والبرمجي. فهو ليس مجرد تطبيق برمجي، بل هو أداة بيداغوجية (تعليمية) تسد فجوة حقيقية في عملية تدريس المفاهيم المجردة. من خلال توفير بيئة تفاعلية وآمنة للتجربة، يمكّن المشروع الطلاب من بناء فهم حدسي وعميق لمبادئ الحوسبة التي تشكل أساس كل التقنيات الرقمية الحديثة.

---

٩. الملاحق

٩,١ لقطات من واجهة المستخدم

(في هذا القسم، يتم إدراج صور لواجهة البرنامج تظهر محاكاة كل آلة)

مثال: واجهة آلة الدفع للأسفل (PDA)

(هذا رابط وهمي، يجب استبداله بلقطة شاشة حقيقية)

٩. بالتأكيد. بصفتي شريكك في البرمجة، يسعدني أن أقدم لك مجموعة من حالات الاختبار (Test Cases) المصممة للتحقق من صحة وظائف المحاكي لكل نوع من الآلات. يمكنك استخدام هذه المدخلات لضمان أن النظام يعمل كما هو متوقع.

---

١. مدخلات اختبار للآلات المنتهية (Finite Automata - FA)

هنا، يكون الإدخال هو **تعبير نمطي (Regular Expression)**.

الحالة ١: لغة بسيطة

- **اللغة:** سلاسل تبدأ بـ a تليها b واحدة أو أكثر. (مثال: ab abb abbb ...)
- **التعبير النمطي للإدخال:** ab(b)*
- **سلاسل اختبار مقبولة (Accepted):**
  - ab

- abb
- abbbb

- **سلاسل اختبار مرفوضة (Rejected):**
  - a (تحتاج إلى b بعدها)
  - b (يجب أن تبدأ بـ a)
  - ba (ترتيب خاطئ)
  - aba (الـ a الأخيرة غير مسموح بها)

الحالة ٢: لغة تحتوي على اتحاد ونجمة كلين

- **اللغة:** أي سلسلة مكونة من a و b (بأي عدد وترتيب) ولكنها يجب أن تنتهي بالحرف c.
- **التعبير النمطي للإدخال:** c*(a|b)
- **سلاسل اختبار مقبولة (Accepted):**
  - c (سلسلة فارغة من a b تليها c)
  - ac
  - bc
  - aabc
  - babac
- **سلاسل اختبار مرفوضة (Rejected):**
  - a (لا تنتهي بـ c)
  - b
  - ab
  - ca (الـ c ليست في النهاية)
  - acb (تنتهي بـ b)

---

٢. مدخلات اختبار لآلات الدفع للأسفل (Pushdown Automata - PDA)

هنا، يكون الإدخال هو **مجموعة من قواعد الانتقال**.

الحالة ١: اللغة حرة السياق (L={anbn|n≥١})

- **اللغة:** عدد متساوٍ من a يليهم عدد متساوٍ من b.
- **قواعد الانتقال للإدخال:**
- @accept: q2

- q0 a Z q0 AZ
- q0 a q0 AA
- q0 b A q1 e
- q1 b A q1 e
- q1 e Z q2 e
- **سلاسل اختبار مقبولة (Accepted):**
  - ab
  - aabb
  - aaabbb
- **سلاسل اختبار مرفوضة (Rejected):**
  - aab (عدد غير متساوٍ)
  - abb (عدد غير متساوٍ)
  - ba (ترتيب خاطئ)
  - aba (رمز a بعد b)
  - e (السلسلة الفارغة، لأن n≥١)

الحالة ٢: اللغة حرة السياق (المتناظرات - Palindromes)

- **اللغة:** الكلمات المتناظرة حول الحرف c مثل wcw^R حيث w هي أي سلسلة من a و b.
- **قواعد الانتقال للإدخال:**
- @accept: q2
- q0 a Z q0 aZ
- q0 b Z q0 bZ
- q0 a q0 aa
- q0 b a q0 ba
- q0 a b q0 ab
- q0 b q0 bb
- q0 c Z q1 Z
- q0 c a q1 a
- q0 c b q1 b
- q1 a q1 e

- q1 b q1 e
- q1 e Z q2 e
- **سلاسل اختبار مقبولة (Accepted):**
  - c
  - aca
  - bcb
  - abccba
  - babcab
- **سلاسل اختبار مرفوضة (Rejected):**
  - acb (النصف الثاني ليس معكوس الأول)
  - bca
  - abc
  - cca (يجب وجود c واحدة فقط)

---

٣. مدخلات اختبار لآلة تورنغ (Turing Machine - TM)

هنا، يكون الإدخال هو **مجموعة من قواعد الانتقال**. ملاحظة: يفترض المحاكي (حسب الكود) أن الحالة ذات الرقم الأعلى هي حالة القبول، والتي تليها هي حالة الرفض.

الحالة ١: اللغة حساسة للسياق (L={anbncn|n≥١})

- **اللغة:** عدد متساوٍ من a ثم b ثم c.
- **الخوارزمية:** تحول a إلى X ثم تبحث عن b وتحولها إلى Y ثم تبحث عن c وتحولها إلى Z ثم تعود إلى البداية وتكرر.
- **قواعد الانتقال للإدخال:**
- // q0: Start. Find 'a' -> X go q1. If 'Y' found all a's are done go to verification q4.
- q0 a q1 X R
- q0 Y q4 Y R
-
- // q1: Find 'b'. Skip 'a's and 'Y's. Change 'b' to 'Y' go q2.
- q1 a q1 a R
- q1 Y q1 Y R
- q1 b q2 Y R
-

- // q2: Find 'c'. Skip 'b's. Change 'c' to 'Z' go q3 (rewind).
- q2 b q2 b R
- q2 Z q2 Z R
- q2 c q3 Z L
-
- // q3: Rewind left over all symbols until we find the start 'X' then move right and go to q0.
- q3 a q3 a L
- q3 b q3 b L
- q3 Y q3 Y L
- q3 Z q3 Z L
- q3 X q0 X R
-
- // q4: Verification state. Scan right over Y's and Z's. If you find a blank accept (go to q5).
- q4 Y q4 Y R
- q4 Z q4 Z R
- q4 _ q5 _ R
-
- // q5 will be the ACCEPT state as it's the highest number.
- **سلاسل اختبار مقبولة (Accepted):**
  - abc
  - aabbcc
- **سلاسل اختبار مرفوضة (Rejected):**
  - abbc
  - aabbc
  - abcc
  - acb (ترتيب خاطئ)

الحالة ٢: ناسخ السلاسل

- **اللغة:** تحويل سلسلة من a و b (مثل aab) إلى aab_aab.

- **الخوارزمية:** تعلم كل رمز، اذهب إلى النهاية واكتبه، ثم عد. استخدم الأحرف الكبيرة لتمييز الرموز التي تمت معالجتها.
- **قواعد الانتقال للإدخال:**
- // q0: Find first 'a' or 'b'. Change to uppercase and go to corresponding state (q1 for a q2 for b). If all are uppercase go to cleanup q3.
- q0 a q1 A R
- q0 b q2 B R
- q0 _ q3 _ L // End of first pass begin cleanup.
-
- // q1: Found 'a'. Go to the end of the tape and write 'a'.
- q1 a q1 a R
- q1 b q1 b R
- q1 A q1 A R
- q1 B q1 B R
- q1 _ q1_write_a a L
-
- // q1_write_a: Rewind back to the first uppercase letter.
- q1_write_a a q1_write_a a L
- q1_write_a b q1_write_a b L
- q1_write_a A q0 A R
- q1_write_a B q0 B R
-
- // q2: Found 'b'. Go to the end of the tape and write 'b'. (Similar to q1)
- q2 a q2 a R
- q2 b q2 b R
- q2 A q2 A R
- q2 B q2 B R
- q2 _ q2_write_b b L
-
- // q2_write_b: Rewind.
- q2_write_b a q2_write_b a L

- q2_write_b b q2_write_b b L
- q2_write_b A q0 A R
- q2_write_b B q0 B R
-
- // q3: Cleanup. Go left changing uppercase back to lowercase.
- q3 A q3 a L
- q3 B q3 b L
- q3 _ q4 _ R // All done. Halt and accept.
-
- // q4 is the ACCEPT state.

هذا المثال معقد وقد يتطلب ترقيمًا مختلفًا للحالات ليعمل بشكل صحيح مع المحاكي. لنستخدم مثال أبسط.

الحالة ٣ (بديل أبسط): حذف كل الـ 'b' من السلسلة

- **اللغة:** تأخذ سلسلة من a و b وتحذف جميع الـ b.
- **الخوارزمية:** امسح الشريط. إذا قرأت a فاتركه. إذا قرأت b فاستبدله بالرمز الفارغ _ ثم قم بإزاحة باقي السلسلة إلى اليسار (هذه عملية معقدة). **خوارزمية أسهل:** امسح الشريط. إذا قرأت a فاكتب a وتحرك يمينًا. إذا قرأت b فتحرك يمينًا دون كتابة شيء (هذا يتطلب شريطين). **الخوارزمية الأبسط للتنفيذ هنا:** استبدل كل b برمز X ثم عد وانسخ فقط الـ a إلى نهاية الشريط.
- **دعنا نستخدم خوارزمية أبسط: استبدال a بـ b والعكس.**
- // q0: Start scanning right.
- q0 a q0 b R
- q0 b q0 a R
- q0 _ q1 _ L // End of string move to accept state q1.
-
- // q1 is the ACCEPT state.
- **سلسلة إدخال:** aabba
- **النتيجة المتوقعة على الشريط:** bbaab
- **سلسلة إدخال:** abab
- **النتيجة المتوقعة على الشريط:** baba

---

١٠. المراجع

- Motwani R. Hopcroft J. E. & Ullman J. D. (٢٠٠٦). *Introduction to Automata Theory Languages and Computation*. Addison-Wesley.
- Sipser M. (٢٠١٢). *Introduction to the Theory of Computation*. Cengage Learning.
- توثيق مكتبة .NET Framework لـ System.Windows.Forms و System.Drawing.

في البداية يتم تعريف كلاسات لكل نوع من أنواع الالات لاجل تحقيق مبدأ الـ oop الذي تم اخذه سابقاً وايضاً للحصول على الكفاءة المرجوة عند التنفيذ وكانت الكلاسات كلتالي :

١- FaLogic.cs
٢- PdaLogic.cs
٣- TmLogic.cs
٤- AutomataModels.cs
٥- VisualState لتمثيل أي حاله مرئية في الرسم البياني

في المرحلة التالية يتم تعريف المكاتب لاستخدام الـ Methods الخاصة بها فيForm الـ Maintest وهي كالتالي :

```
using System
using System.Collections.Generic
using System.Drawing
using System.Drawing.Drawing2D
using System.Linq
using System.Text.RegularExpressions
using System.Threading
using System.Threading.Tasks
using System.Windows.Forms
```

بعد ذالك يتم تعريف المتغيرات المطلبة لانجاز المشروع وهي كالتالي :

```
  private RadioButton rbFiniteAutomata  rbPushdownAutomata
rbTuringMachine
  private RadioButton rbDeterministic  rbNondeterministic
  private Panel mainContentPanel
  private ToolTip
```

الكلاس الخاص بتمثيل الرسم البياني لاي حالة

```
public class VisualState
 {
     public int Id { get   set   }
     public Point Position { get   set   }
     public bool IsAcceptState { get   set   }
     public bool IsRejectState { get   set   } //  خاص بآلة turing
 }
```

اما هنا فهو الكلاس الخاص بـ state أي الحالة ولدية خصائصة المعينة المفهومة من الكود المرفق :

```
public class State
 {
     private static int nextId = 0
     public int Id { get   }
     public bool IsAcceptState { get   set   }
     public Dictionary<char  List<State>> Transitions { get   } = new Dictionary<char  List<State>>()
     public Point Position { get   set   }
     public string DfaStateIdentifier { get   set   }
     public State(bool isAccept = false) { this.Id = nextId++   this.IsAcceptState = isAccept   }
     public void AddTransition(char symbol  State toState) { if (!Transitions.ContainsKey(symbol)) {
Transitions[symbol] = new List<State>()   } Transitions[symbol].Add(toState)   }
     public static void ResetIdCounter() => nextId = 0
 }
```

الكلاس الخاص بـ NFA Formte الذي يحدد الخصائص لنقطة البداية والنهاية

```
        public class NfaFragment { public State Start { get   set   }
public State End { get   set   } }
```

الكلاسات الخاصة للـ PDA Models

وهنا الكلاس PDATransitionKey الذي يمثل مفتاحًا فريدًا لتحديد انتقال معين في آلة الأوتوماتا بالدفع للأسفل(PDA)

تتكون من الحالة الحالية FromStateId الرمز الذي يتم قراءته من الإدخال InputSymbol والرمز الذي يتم إزالته من أعلى المكدس StackPopSymbol

```
public class PDATransitionKey
{
    public int FromStateId { get   set   }
    public char InputSymbol { get   set   }
    public char StackPopSymbol { get   set   }

    public override bool Equals(object obj)
    {
        if (!(obj is PDATransitionKey other)) return false
        return FromStateId == other.FromStateId &&
               InputSymbol == other.InputSymbol &&
               StackPopSymbol == other.StackPopSymbol
    }

    public override int GetHashCode()
    {
        unchecked
        {
            int hash = 17
            hash = hash * 23 + FromStateId.GetHashCode()
            hash = hash * 23 + InputSymbol.GetHashCode()
            hash = hash * 23 + StackPopSymbol.GetHashCode()
            return hash
        }
```

اما الكلاس التالي هو PDATransition والذي يمثل قاعدة انتقال واحدة للالة الـ PDA وعملة الرايسي هو لتحديد سلوك آلة الـPDA حيث نعرف جميع الحركات الممكنة للآلة

```
public class PDATransition
 {
     public int FromStateId { get   set   }
     public char InputSymbol { get   set   } // '\0' ـل ε
     public char StackPopSymbol { get   set   } // '\0' ـل ε
     public int NextStateId { get   set   }
     public string StackPushSymbols { get   set   } // string.Empty ـل ε
 }
```

اما الكلاس PushdownAutomaton والذي يمثل بعملة نموذج آلة أوتوماتا بالدفع للأسفل (PDA) بأكمله واهميته تتمحور في التمثيل الكامل لآلة الـPDA وتجمع كل مكوناتها الهيكلية اللازمة للمحاكاة

```
public class PushdownAutomaton
 {
     public Dictionary<int  List<PDATransition>> Transitions { get   } = new
Dictionary<int  List<PDATransition>>()
     public int StartStateId { get   set   }
     public HashSet<int> AcceptStates { get   } = new HashSet<int>()
     public char StartStackSymbol { get   set   } = 'Z'
 }
```

اما كلاس الـ PDAConfiguration فهو يمثل حالة معينة لآلة الـ PDA أثناء المحاكاة وتكمن الأهمية لهذا الكلاس في التقاط لقطة لآلة الـ PDA في لحظة معينة أثناء تشغيلها وهي ضرورية لتتبع مسار المحاكاة وللتعامل مع الآلات غير الحتمية

```
 public class PDAConfiguration
{ public int CurrentStateId { get   set   } public int InputPointer { get   set
} public Stack<char> MachineStack { get   set   }
public List<string> TraceHistory { get   set   } }
```

اما كلاس الـ PdaSimulationResult فيعمل على تحمل نتيجة محاكاة آلة الـ PDA على سلسلة إدخال معينة وتكمن الأهمية في توفر ملخصًا شاملاً لنتائج المحاكاة بما في ذلك ما إذا كانت السلسلة مقبولة ولماذا بالإضافة إلى سجل تفصيلي للخطوات

```
public class PdaSimulationResult { public bool IsAccepted { get   } public
List<string> Trace { get   } public bool IsDeterministicViolation { get   }
public PdaSimulationResult(bool accepted  List<string> trace  bool violation) {
IsAccepted = accepted   Trace = trace   IsDeterministicViolation = violation   }
}
```

الان مع الكلاسات الخاصة بـ turing machin :

الكلاس الأول والذي بعمل على تحديد الاتجاهات المحتملة التي يمكن أن يتحرك فيها رأس آلة تورينج على الشريط

```
        public enum TapeMove { L  R S }
```

الكلاس الثاني TMTransitionKey يمثل على الهيكل مفتاحًا فريدًا لتحديد انتقال معين في آلة تورينج تتكون من الحالة الحالية StateIdوالرمز الذي تتم قراءته من الشريط ReadSymbol ومن الأهمية لهذا الكلاس انه يستخدم كمفتاح في قاموس Transitionsلآلة تورينج لربط زوج (حالة، رمز مقروء) بانتقال واحد محدد مثل PDATransitionKey توفر طرق Equals و GetHashCode للتعامل معها كمفتاح قاموس

```
public struct TMTransitionKey { public readonly int StateId   public
readonly char ReadSymbol   public TMTransitionKey(int stateId  char
readSymbol) { StateId = stateId   ReadSymbol = readSymbol   } public
override bool Equals(object obj) => obj is TMTransitionKey other &&
this.StateId == other.StateId && this.ReadSymbol == other.ReadSymbol
public override int GetHashCode() { unchecked { return (StateId *
397) ^ ReadSymbol.GetHashCode()   } } }
```

الكلاس الثالث TMTransition يمثل قاعدة انتقال واحدة لآلة تورينج تحدد

١- الحالة التي تنتقل إليها الآلة NextStateId
٢- الذي يكتب على الشريط في الخلية الحالية WriteSymbol
٣- MoveDirection الاتجاه الذي يتحرك فيه رأس الشريط بعد الانتقال يستخدم TapeMove للقيم (L R S)

وتاتي أهمية هذه الاله في تحديد سلوك آلة تورينج حيث تعرف جميع الحركات الممكنة للآلة بناءً على حالتها الحالية والرمز المقروء.

```
        public class TMTransition { public int NextStateId { get
set   } public char WriteSymbol { get   set   } public TapeMove
MoveDirection { get   set   } }
```

الكلاس الرابع TuringMachine والذي يعمل على اله تيورنج بأكملها

أهميتها هي التمثيل الكامل لآلة تورينج وتجمع كل مكوناتها الهيكلية اللازمة للمحاكاة

```
        public class TuringMachine { public
Dictionary<TMTransitionKey  TMTransition> Transitions { get   set   }
= new Dictionary<TMTransitionKey  TMTransition>()   public int
StartStateId { get   set   } public int AcceptStateId { get   set   }
public int RejectStateId { get   set   } }
```

الى هنا نستطيع القول باني الكلاسات المستخدمة في المشروع انتهت الان ننتقل الى داله الأساس التي سيبدا المشروع التنفيذ من عندها وهي الـ constractor

داله المهيئ التي سوف ينطلق المشروع من خلالها والتي تحوي دوال معينة الدوال التي تتعلق بشكل أساسي ببناء وتحديث واجهة المستخدم (UI) لتطبيق المحاكاة سوف يتم شرحها لاحقاً اما خصائص الـFORM فهي معروفة

```
public testMain()
 {
     SetupUI()              // يتم استدعاء SetupUI أولاً الآن
     UpdateActivePanel()    // ثم يتم تحديث اللوحة بعد تهيئتها
     this.Text = "(مع الرسوم البيانية) 2.0 الإصدار – محاكي النظرية الاحتسابية"
     this.MinimumSize = new Size(1200  800)
     this.Size = new Size(1300  850)
 }
```

دالة الـ SetupUI فهذه الداله التي تنفذ ببداية اقلاع المشروع تتضمن التالي :

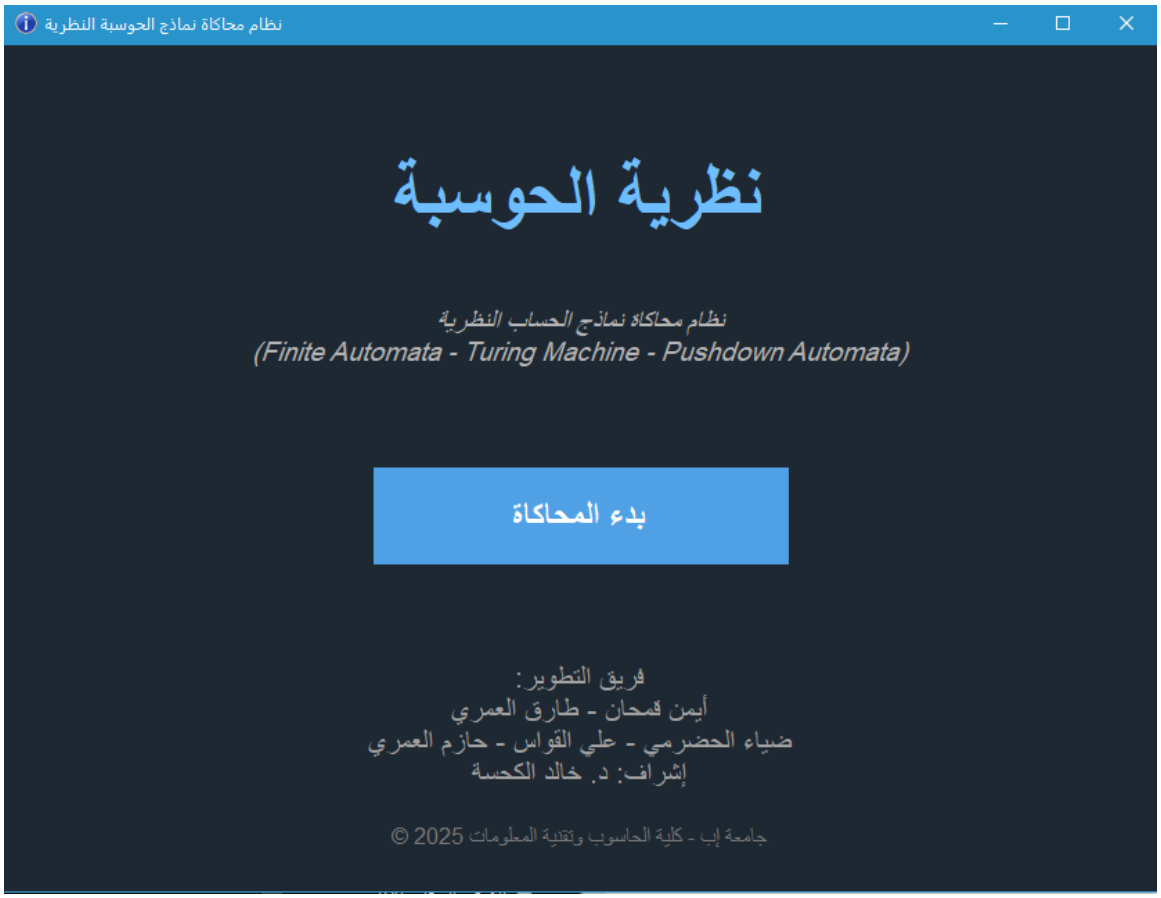
- **العمل :** هذه الدالة هي المسؤولة عن بناء وتكوين جميع عناصر واجهة المستخدم الرئيسية عند بدء تشغيل التطبيق. تقوم بـ :

- تعيين لون الخلفية.
- تهيئة ToolTip لعرض تلميحات عند تمرير الماوس.
- إنشاء لوحات تنظيمية مثل TableLayoutPanel وFlowLayoutPanel لترتيب العناصر بشكل صحيح.
- إنشاء أزرار الراديو (RadioButton) لاختيار نوع الآلة Finite Automata Pushdown Automata Turing Machine ونوعها الفرعي Deterministic Non-deterministic
- ربط أحداث CheckedChanged لأزرار الراديو بدالة OnMachineTypeChanged والتي من المفترض أن تستدعي UpdateActivePanel
- إضافة جميع هذه العناصر إلى النافذة الرئيسية.((this.Controls.Add(mainLayout)

- **أهميتها :** هي نقطة البداية لإنشاء المظهر العام للتطبيق وتنظيم الأجزاء الأساسية من الواجهة التي تسمح للمستخدم بالاختيار بين أنواع الآلات المختلفة.

```
        private void SetupUI()
        {
            this.BackColor = Color.FromArgb(240  240  240)
            toolTip = new ToolTip()
            var mainLayout = new TableLayoutPanel { Dock = DockStyle.Fill  ColumnCount = 1  RowCount = 2 }
            mainLayout.RowStyles.Add(new RowStyle(SizeType.AutoSize))
            mainLayout.RowStyles.Add(new RowStyle(SizeType.Percent  100f))
            var selectionPanel = new FlowLayoutPanel { Dock = DockStyle.Top  AutoSize = true  Padding = new Padding(5)
WrapContents = false }
            var gbMachineType = new GroupBox { Text = "نوع الآلة"  AutoSize = true }
            var machineTypeLayout = new FlowLayoutPanel { FlowDirection = FlowDirection.TopDown  AutoSize = true }
            rbFiniteAutomata = new RadioButton { Text = "آلات منتهية (FA)"  Checked = true  AutoSize = true }
            rbPushdownAutomata = new RadioButton { Text = "آلات الدفع للأسفل (PDA)"  AutoSize = true }
            rbTuringMachine = new RadioButton { Text = "آلة تورنغ (TM)"  AutoSize = true }

            rbFiniteAutomata.CheckedChanged += OnMachineTypeChanged
            rbPushdownAutomata.CheckedChanged += OnMachineTypeChanged
            rbTuringMachine.CheckedChanged += OnMachineTypeChanged

            machineTypeLayout.Controls.AddRange(new Control[] { rbFiniteAutomata  rbPushdownAutomata  rbTuringMachine })
            gbMachineType.Controls.Add(machineTypeLayout)
            var gbSubType = new GroupBox { Text = "النوع الفرعي"  AutoSize = true }

            var subTypeLayout = new FlowLayoutPanel { FlowDirection = FlowDirection.TopDown  AutoSize = true }

            rbDeterministic = new RadioButton { Text = "محدودة (Deterministic)"  Checked = true  AutoSize = true }
            rbNondeterministic = new RadioButton { Text = "غير محدودة (Non-deterministic)"  AutoSize = true }

            subTypeLayout.Controls.AddRange(new Control[] { rbDeterministic  rbNondeterministic })
            gbSubType.Controls.Add(subTypeLayout)
            selectionPanel.Controls.AddRange(new Control[] { gbMachineType  gbSubType })
            mainContentPanel = new Panel { Dock = DockStyle.Fill  Padding = new Padding(5) }   // تم تعريفها هنا الآن
            mainLayout.Controls.AddRange(new Control[] { selectionPanel  mainContentPanel })
            this.Controls.Add(mainLayout)
        }
```

ويبدو الواجهة للمشروع الرسومي كلتالي



والذي توضح لنا سير البرنامج حيث وان الـ radiobutton هي التي تحدد ما النوع المستهدف لتعالجة علماً بان لكل اله panelsخاص بها

الان في ال dfa في حقل الادخال للتعبير المنطقي لرسم الاله عند ادخال مثلاً (a|b)*b والضغط على الزر بناء الاله فانه سوف يرسم الاله الخاصه بالـتعبير المدخل في الحقل وينتج لنا انتقالات الحالات في الـ panelالمجاور لـ panel الرسم

بعد ذلك استطيع اختبار السلسلة ا لمدخلة في الحقل الخاص ب باختبار السلسلة وعند الضغط على اختبار سوق يقرر هل السلسلة المدخلة مقبولة ام لا .

الدالة الأخرى التي ايضاً توجد في المشيد UpdateActivePanel والتي تعمل على التالي :

- **العمل :**تُستدعى هذه الدالة عندما يغير المستخدم نوع الآلة (FA PDA TM) أو عندما تتغير بعض الخصائص (مثل الحتمية). تقوم بـ :
  - مسح أي محتوى سابق من اللوحة الرئيسية للمحتوى(mainContentPanel)
  - تعطيل خيار غير محدودة (rbNondeterministic.Enabled = false) إذا تم تحديد آلة تورينج لأن آلات تورينج عادةً ما تُعتبر حتمية في هذا السياق.
  - تحديد أي لوحة فرعية يجب عرضها بناءً على زر الراديو المحدد حاليًا (FA PDA TM) عن طريق استدعاء دوال مثل CreateFiniteAutomataPanel() أو CreatePushdownAutomataPanel() CreateTuringMachinePanel()
- **أهميتها :**هي الدالة التي تتحكم في ديناميكية الواجهة، حيث تضمن عرض الجزء الصحيح من الواجهة للمستخدم بناءً على اختياره لنوع الآلة وتضبط الخيارات المتاحة بشكل صحيح.

```
        private void UpdateActivePanel()
        {
            mainContentPanel.Controls.Clear()
            rbNondeterministic.Enabled = !rbTuringMachine.Checked
            if (rbTuringMachine.Checked) rbDeterministic.Checked = true

            if (rbFiniteAutomata.Checked)
mainContentPanel.Controls.Add(CreateFiniteAutomataPanel())
            else if (rbPushdownAutomata.Checked)
mainContentPanel.Controls.Add(CreatePushdownAutomataPanel())
            else if (rbTuringMachine.Checked)
mainContentPanel.Controls.Add(CreateTuringMachinePanel())
        }
```

الان دالة الـ CreateFiniteAutomataPanel والذي تخص اله turing :

العمل :تقوم هذه الدالة بإنشاء وتكوين جميع عناصر واجهة المستخدم الخاصة بـ **الآلات المتناهية - Finite Automata FA** تتضمن عناصر مثل :

- مربع نص لإدخال التعبير النمطي.(regexInput)
- زر "بناء الآلة.(buildFaButton) "
- لوحة لرسم مخطط الآلة.(faDiagramPanel)
- جدول لعرض انتقالات الآلة.(faTransitionTable)
- مربع نص لاختبار السلاسل (faTestStringInput) وزر "اختبر.(faTestButton) "
- ملصق لعرض نتيجة الاختبار.(faResultLabel)
- ربط الأحداث المناسبة مثل Click للأزرار و Paintللوحة الرسم.

أهميتها :تُنشئ جزء الواجهة المخصص لمحاكاة وتصوير الآلات المتناهية، مما يوفر للمستخدم جميع الأدوات اللازمة للتعامل مع هذا النوع من الآلات

الان دالة الـ CreatePushdownAutomataPanel وهي التي تمتلك المكدس لحفض العمليات وتعمل كالتالي :

العمل :تقوم هذه الدالة بإنشاء وتكوين جميع عناصر واجهة المستخدم الخاصة بـ **آلات الدفع للأسفل Pushdown Automata - PDA** تتضمن عناصر مثل :

- مربع نص كبير لإدخال تعريفات انتقالات الـ PDA (pdaTransitionsInput) مع تلميح للصيغة.
- لوحة لرسم مخطط الـ PDA (pdaDiagramPanel).
- زر بناء الآلة buildPdaButto (PDA)
- مربع نص لاختبار السلاسل (pdaTestStringInput) وزر اختبر(testPdaButton)
- ملصق لعرض نتيجة الاختبار.(pdaResultLabel)
- صندوق قائمة لعرض سجل التتبع.(pdaTraceLog)
- جدول لعرض انتقالات الـ PDA (pdaTransitionTable).
- ربط الأحداث المناسبة مثل Click للأزرار و Paint للوحة الرسم.

أهميتها :تُنشئ جزء الواجهة المخصص لمحاكاة وتصوير آلات الدفع للأسفل، مما يوفر للمستخدم جميع الأدوات اللازمة لتعريفها واختبارها وتتبع محاكاتها.

الشكل الخاص بالة الـ PDA

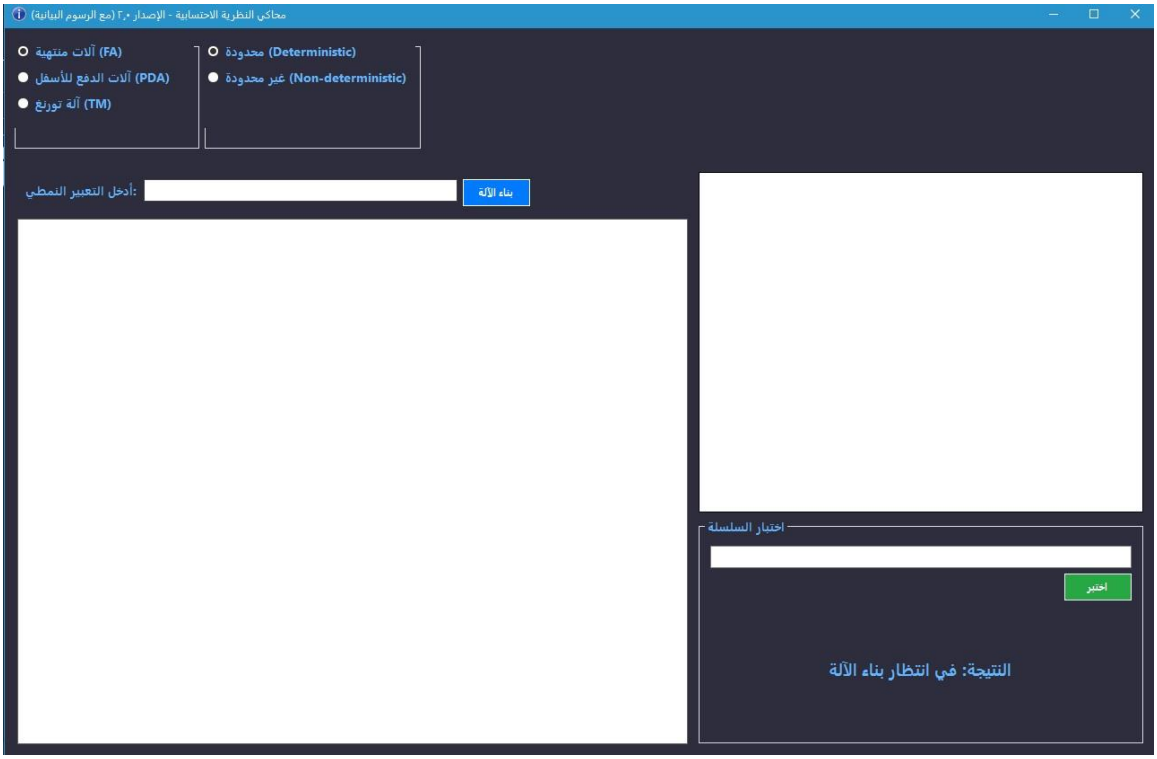
الان دالة الـ CreateTuringMachinePanel الخاضة بالالة turing macchin

العمل :تقوم هذه الدالة بإنشاء وتكوين جميع عناصر واجهة المستخدم المخصصة لـ **آلات تورينج Turing Machines - TM** تتضمن عناصر مثل :

- مربع نص كبير لإدخال تعريفات انتقالات آلة تورينج(tmTransitionsInput) ، مع تلميح يوضح الصيغة المطلوبة للإدخال.
- لوحة لرسم الرسم البياني لحالات آلة تورينج(tmDiagramPanel) ، مع تفعيل خاصية التمرير التلقائي (AutoScroll) لتسهيل عرض المخططات الكبيرة.
- زر بناء الآلة(buildTmButton) (TM)
- مربع نص لإدخال سلسلة الاختبار (testTmInput) وزر "شغل (testTmButton) "لبدء المحاكاة.
- ملصق لعرض حالة ونتيجة المحاكاة.(tmResultLabel)
- لوحة لتمثيل الشريط بصريًا(tmTapeVisualizer) ، حيث سيتم عرض حالة الشريط وموقع الرأس أثناء المحاكاة.
- جدول لعرض انتقالات آلة تورينج.(tmTransitionTable)
- ربط الأحداث المناسبة مثل Click للأزرار و Paint للوحات الرسم والشريط.

أهميتها :تُنشئ واجهة المستخدم الكاملة التي تسمح للمستخدم بتعريف آلة تورينج، عرض مخططها، إدخال سلاسل للاختبار، تشغيل المحاكاة، ومشاهدة حالة الشريط ونتائج التشغيل.

الشكل العالم لاله الـ turing machin



الان دالة الـ OnMachineTypeChanged والتي تعتمل على تحديد الـ radiiobutton الذي تم الضغط علية عشان يقرر أي واجهه يفتح كما ان هذه الداله ترتبط بدوال انشاء الالات المذكوره سايقاً كمثال توضيحي بهذا الشكل

```
rbFiniteAutomata.CheckedChanged += OnMachineTypeChanged
```

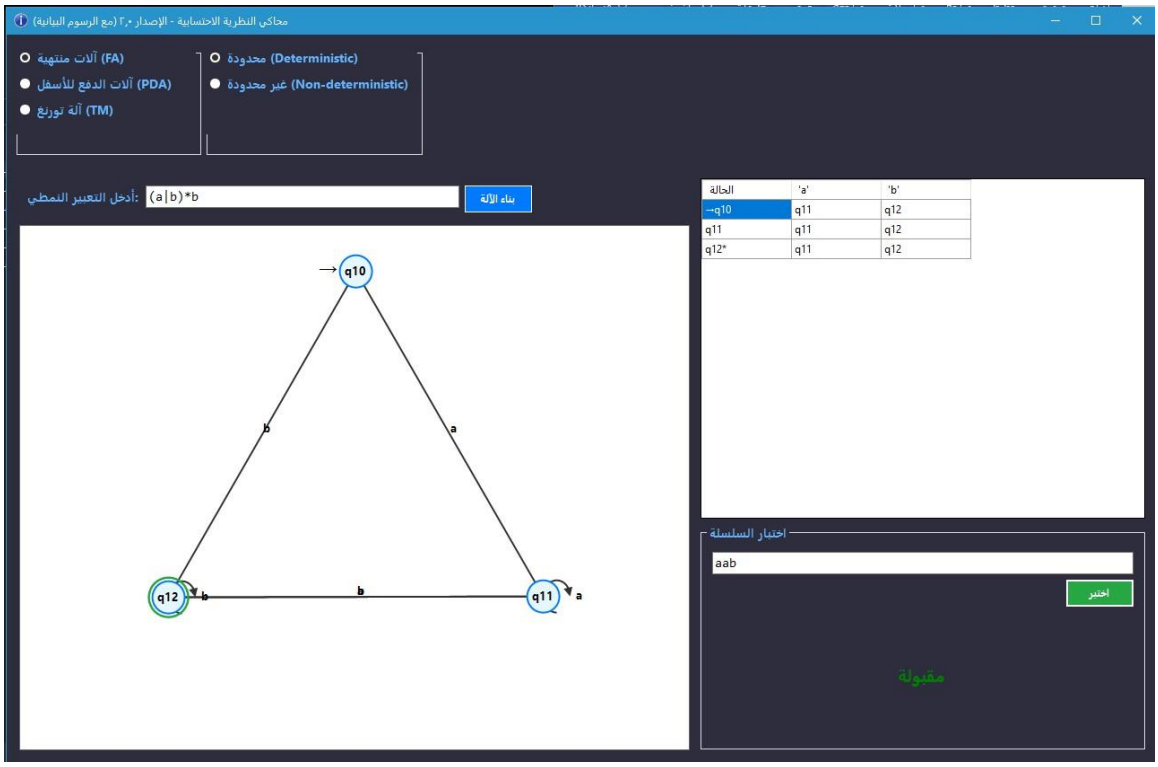
دالة الـ BuildFaButton_Click وهي المسؤلة على تنفيذ بناء او رسم اله الـ DFA

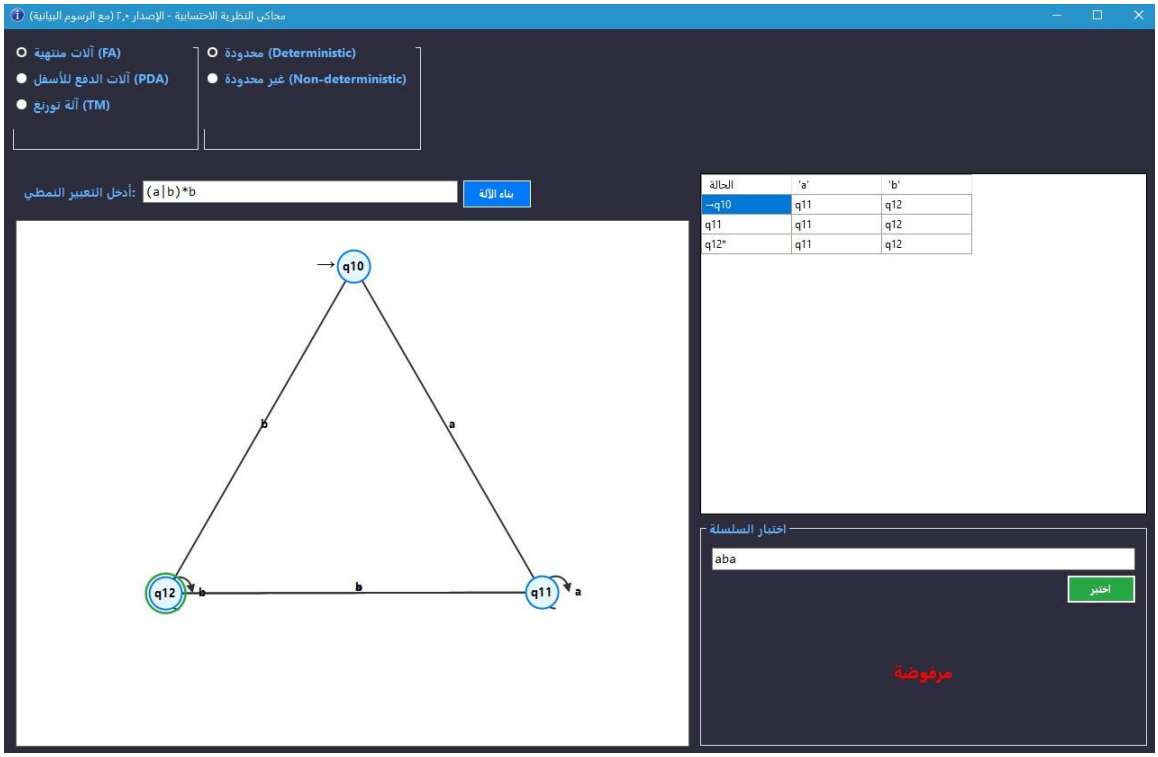
العمل :تُستدعى هذه الدالة عند النقر على زر "بناء الآلة" الخاص بالآلات المتناهية. تقوم بالخطوات التالية :

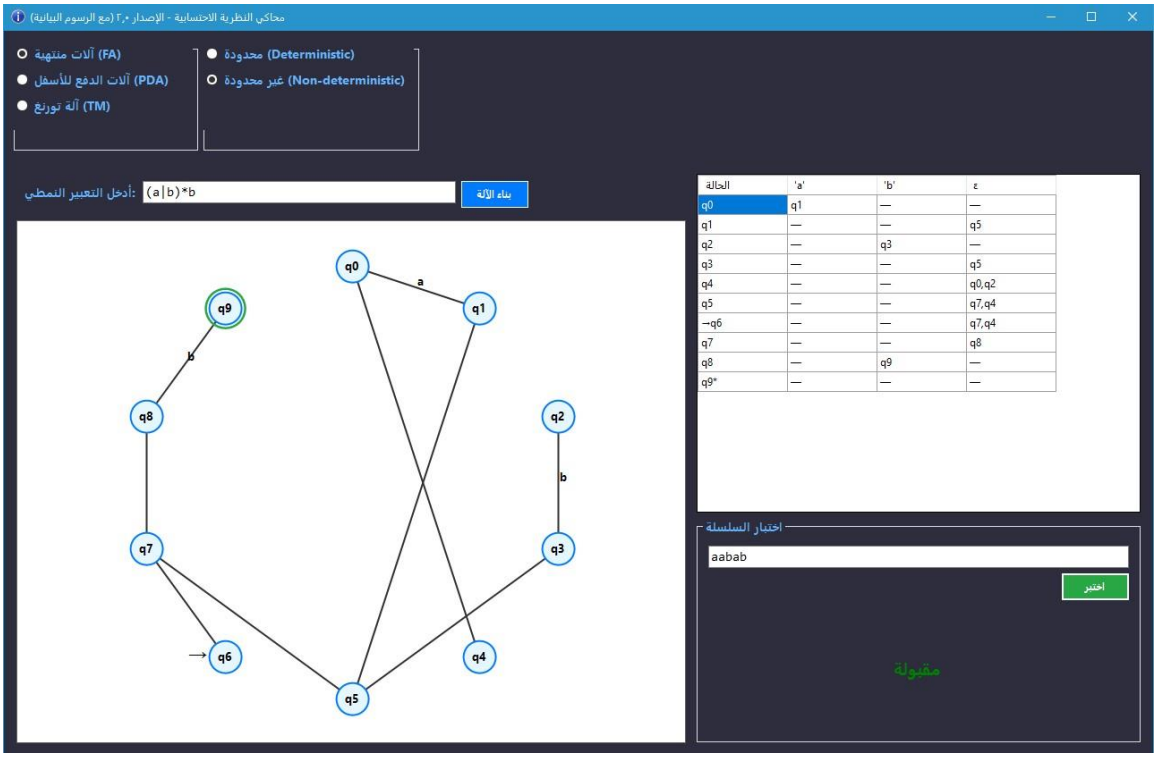
- تعيد تعيين عداد مُعرفات الحالات (State.ResetIdCounter()) لضمان بدء العد من الصفر لكل آلة جديدة.
- تضيف علامات الترتيب المناسبة) مثل عامل الدمج (implicit concatenation إلى التعبير النمطي المدخل (regexInput.Text).
- تحول التعبير النمطي من صيغة infix إلى.(InfixToPostfix) postfix
- بناءً على ما إذا كان المستخدم قد اختار "محدودة "(Deterministic) أو "غير محدودة(Non-deterministic)
  - **إذا كانت حتمية :(DFA)** تبني NFA من التعبير النمطي، ثم تحول هذه الـ NFA إلى DFA (Non-deterministic Finite Automaton)إلى Deterministic Finite Automaton). ثم تقوم بتحديث جدول الانتقالات في الواجهة (UpdateFaTransitionTable).
  - **إذا كانت غير حتمية :(NFA)** تبني NFA مباشرة من التعبير النمطي وتستخدمها. ثم تقوم بتحديث جدول الانتقالات في الواجهة لعرض.(UpdateNfaTransitionTable) NFA
- تضبط مواضع الحالات في لوحة الرسم (PositionStates) وتطلب إعادة رسم اللوحة (faDiagramPanel.Invalidate()) لعرض الآلة الجديدة.
- تحدث رسالة الحالة في الواجهة لتشير إلى أن الآلة جاهزة.
- تحتوي على كتلة try-catchللتعامل مع أي أخطاء قد تحدث أثناء عملية البناء.

أهميتها : هي الدالة الرئيسية التي تبدأ عملية تحويل التعبير النمطي الذي أدخله المستخدم إلى نموذج آلة متناهية FA أوNFA وعرضها في الواجهة.

الرسم الخاص باله الـ DFA







دالة TestDfaString(string input)

- **العمل :**تحاكي هذه الدالة تشغيل آلة DFA (الآلة المتناهية الحتمية) على سلسلة إدخال معينة .
  - تبدأ من الحالة الابتدائية.
  - لكل رمز في سلسلة الإدخال، تحاول الانتقال إلى الحالة التالية بناءً على هذا الرمز.
  - إذا لم تجد انتقالًا لرمز معين، فإن السلسلة تُرفض فورًا.
  - بعد معالجة جميع الرموز، تتحقق مما إذا كانت الآلة في حالة قبول.
  - تحديث faResultLabelبالنتيجة (مقبولة أو مرفوضة) وباللون المناسب.
- **أهميتها :**تُنفذ منطق محاكاة الآلات المتناهية الحتمية.

دالة GetAllStatesFromNfa(State startState)

- **العمل :**هذه الدالة هي دالة مساعدة تقوم بجمع جميع الحالات المتاحة في آلة NFA بدءًا من الحالة الابتدائية المحددة. تستخدم خوارزمية بحث في العمق (DFS) أو بحث في العرض) (BFS) هنا تستخدمStack ، مما يشير إلى (DFS لزيارة جميع الحالات التي يمكن الوصول إليها والانتقالات بينها.
- **أهميتها :**ضرورية للحصول على قائمة كاملة بالحالات لتحديث الجدول ورسم المخطط، بغض النظر عن كون الآلة DFA أو.NFA

دالة UpdateNfaTransitionTable(State startState List<State> allStates)

- **العمل :**تقوم هذه الدالة بتحديث جدول الانتقالات في واجهة المستخدم (faTransitionTable) لعرض انتقالات آلة .NFA
  - تقوم بمسح الأعمدة والصفوف الحالية.

- تضيف أعمدة للحالات ولرموز الإدخال ولـ "ε" (إبسيلون) إذا كانت.NFA
- تملأ الجدول بالانتقالات لكل حالة، مع الإشارة إلى الحالة الابتدائية (→) والحالات المقبولة.(*)

- **أهميتها :**تعرض للمستخدم بنية الـ NFA وانتقالاتها بصريًا في شكل جدولي.

دالة معالجة حدث النقر FaTestButton_Click(object sender EventArgs e)

- **العمل :**تُستدعى هذه الدالة عند النقر على زر "اختبر" الخاص بالآلات المتناهية. تقوم بالتحقق مما إذا كانت الآلة قد تم بناؤها (بواسطة startStateDfa)، ثم تستدعي إما TestDfaStringإذا كانت الآلة حتمية، أو TestNfaStringإذا كانت غير حتمية.
- **أهميتها :**تبدأ عملية اختبار ما إذا كانت سلسلة الإدخال التي قدمها المستخدم مقبولة من قبل الآلة المتناهية المبنية.

دالة TestNfaString(string input)

- **العمل :**تحاكي هذه الدالة تشغيل آلة NFA (الآلة المتناهية غير الحتمية) على سلسلة إدخال معينة .
  - تبدأ بحساب "إغلاق إبسيلون (Epsilon Closure) "للحالة الابتدائية) مجموعة جميع الحالات التي يمكن الوصول إليها من الحالة الابتدائية عبر انتقالات.(ε
  - لكل رمز في سلسلة الإدخال، تقوم بحساب مجموعة الحالات الجديدة التي يمكن الوصول إليها من مجموعة الحالات الحالية بعد قراءة الرمز (باستخدام (Moveثم تحسب إغلاق إبسيلون لتلك المجموعة الجديدة.
  - إذا أصبحت مجموعة الحالات الحالية فارغة في أي نقطة، تُرفض السلسلة.
  - بعد معالجة جميع الرموز، تتحقق مما إذا كانت أي من الحالات الحالية هي حالة قبول.
  - تحديث faResultLabelبالنتيجة واللون.
- **أهميتها :**تُنفذ منطق محاكاة الآلات المتناهية غير الحتمية، والتي تتطلب تتبع مجموعات من الحالات نظرًا لطبيعتها غير الحتمية.

دالة معالجة حدث الرسم FaDiagramPanel_Paint(object sender PaintEventArgs e) هذه الداله تم استدعائها في داله BuildFaButton

- **العمل :**هذه الدالة هي المسؤولة عن رسم مخطط الآلة المتناهية) سواء DFA أو (NFA على لوحة .faDiagramPanel
  - تضبط إعدادات الرسم (مثل SmoothingModeللخطوط الناعمة.(
  - تستخدم فرش وأقلام وألوان مختلفة لرسم :
    - دوائر تمثل الحالات.
    - أسهم تمثل الانتقالات بين الحالات.
    - ملصقات للرموز على الأسهم.
    - علامات خاصة للحالة الابتدائية (→) والحالات المقبولة (دائرة مزدوجة.(
    - حلقات ذاتية (Self-loops) للحالات التي تنتقل إلى نفسها.
  - تتخلص من موارد الرسم بعد الاستخدام.(()Dispose)
- **أهميتها :**تقوم بتحويل التمثيل الداخلي للآلة إلى تمثيل بصري تفاعلي ومفهوم للمستخدم، مما يساعد على فهم بنية الآلة وسلوكها.

PDA الات

ParsePda(string definition bool isDeterministic)

العمل: هذه الدالة مسؤولة عن تحليل (parsing) التعريف النصي لآلة الـ PDA الذي يُدخله المستخدم وتحويله إلى كائن PushdownAutomaton. تقوم بـ:

- قراءة السطور من definition سلسلة نصية تحتوي على تعريف الانتقالات
- استخدام تعبير نمطي (Regex) لمطابقة صيغة كل سطر انتقال مثال q0 a Z q1 AZ
- تحليل الحالة المصدر، رمز الإدخال، رمز الإزالة من المكدس، الحالة الوجهة، ورموز الإضافة إلى المكدس.
- التعامل مع رموز e(إبسيلون) كـ 0\(للمحارف) أو للسلاسل الفارغة
- قراءة الأسطر التي تبدأ بـ accept@ لتحديد حالات القبول.
- التحقق من الحتمية: إذا كانت isDeterministic صحيحة، تتحقق الدالة مما إذا كان التعريف يلتزم بقواعد الآلات الحتمية (DPDA)، أي لا يوجد أكثر من انتقال ممكن لنفس الحالة والرمز المدخل ورمز أعلى المكدس. إذا وجدت انتهاكًا، تُطلق استثناءً.
- تجميع الانتقالات في قاموس ثم إضافتها إلى كائن .PushdownAutomaton أهميتها: هي القلب النابض لإنشاء نموذج الـ PDA القابل للمحاكاة من الإدخال النصي للمستخدم، وتضمن صحة بنية الآلة.

BuildPdaButton_Click(object sender EventArgs e)

العمل: تُستدعى هذه الدالة عند النقر على زر "بناء الآلة ."(PDA) تقوم بـ:

- استدعاء ParsePdaلتحليل نص التعريف وإنشاء كائن .currentPda
- تحديث جدول الانتقالات في الواجهة (pdaTransitionTable) باستخدام .UpdatePdaTransitionTable
- إنشاء قائمة من كائنات VisualState (allPdaStates)لتمثيل حالات الـ PDA بصريًا، مع تحديد حالات القبول.
- تحديد مواضع هذه الحالات على لوحة الرسم (pdaDiagramPanel) وتطلب إعادة رسمها.(()Invalidate)
- تحديث ملصق النتيجة (pdaResultLabel) لتشير إلى أن الآلة جاهزة ومسح سجل التتبع.
- تحتوي على كتلة try-catchلالتقاط وعرض أي أخطاء تحدث أثناء عملية بناء الـ) PDA مثل أخطاء التحليل أو انتهاكات الحتمية
- أهميتها: هي الدالة التي تبدأ عملية بناء وعرض الـ PDA في الواجهة بعد إدخال المستخدم للتعريف.

TestPdaButton_Click(object sender EventArgs e string input)

العمل: تُستدعى هذه الدالة عند النقر على زر "اختبر" لآلة الـ .PDA تقوم بـ:

- مسح سجل التتبع القديم في .pdaTraceLog

- استدعاء الدالة SimulatePdaلتشغيل المحاكاة على سلسلة الإدخال المعطاة، مع الأخذ في الاعتبار ما إذا كان الوضع المحدد هو "حتمي."
- عرض سجل التتبع المستلم من المحاكاة في .pdaTraceLog
- تحديث ملصق النتيجة (pdaResultLabel) ولونه بناءً على ما إذا كانت السلسلة مقبولة، مرفوضة، أو إذا حدث انتهاك للحتمية .
أهميتها: هي الدالة التي تبدأ عملية محاكاة الـ PDA على سلسلة معينة وتعرض نتائج المحاكاة خطوة بخطوة.

SimulatePda(PushdownAutomaton pda string input bool isDeterministicMode)

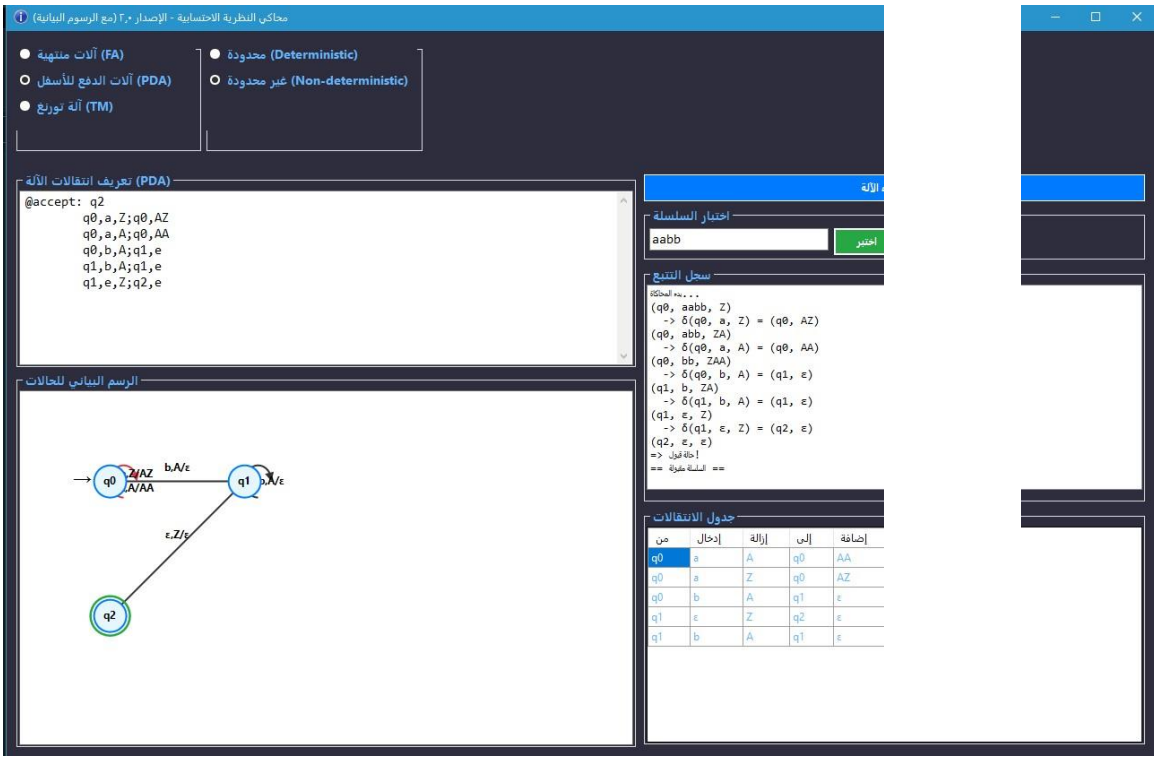
العمل: هذه هي دالة المحاكاة الأساسية لآلة الـ .PDA تطبق خوارزمية البحث في العرض (BFS) باستخدام قائمة انتظار (Queue) لاستكشاف جميع المسارات الممكنة في الـ PDA غير الحتمية.

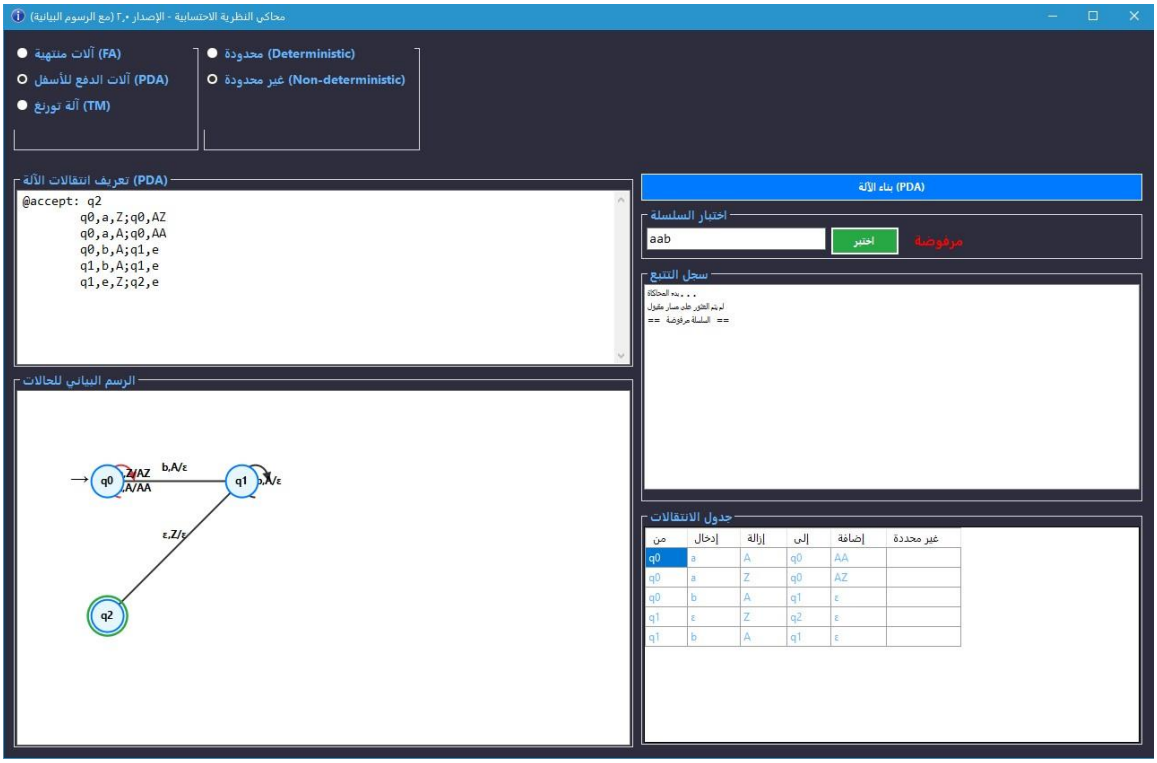
- تبدأ بتهيئة الحالة الأولية للـ) PDA الحالة الابتدائية، مؤشر الإدخال عند ٠، المكدس الأولي.(
- في كل خطوة :
  - تسجل الحالة الحالية للآلة (الحالة، الجزء المتبقي من الإدخال، محتوى المكدس) في سجل التتبع.
  - إذا وصلت إلى نهاية الإدخال وكانت في حالة قبول، تُعلن أن السلسلة مقبولة.
  - تبحث عن جميع الانتقالات الممكنة من الحالة الحالية بناءً على رمز الإدخال الحالي ورمز أعلى المكدس.
  - التحقق من الحتمية: إذا كان isDeterministicModeصحيحًا ووجدت الدالة أكثر من انتقال ممكن لنفس الشروط، فإنها تسجل انتهاكًا للحتمية.
  - لكل انتقال ممكن، تقوم بإنشاء "تكوين (Configuration) "جديد للآلة (حالة جديدة، مؤشر إدخال محدث، مكدس محدث) وتضيفه إلى قائمة الانتظار.
- تتوقف المحاكاة بعد عدد معين من الخطوات (steps < 2000) لتجنب الحلقات اللانهائية.
- تُرجع كائن PdaSimulationResultيحتوي على النتيجة (مقبولة/مرفوضة)، سجل التتبع، وما إذا كان هناك انتهاك للحتمية .
أهميتها: هي المحرك الفعلي الذي يقوم بتشغيل الـ PDA على سلسلة الإدخال، مع القدرة على التعامل مع الطبيعة الحتمية وغير الحتمية للآلة.

PdaDiagramPanel_Paint(object sender PaintEventArgs e)

العمل: هذه الدالة مسؤولة عن رسم مخطط آلة الـ PDA على لوحة .pdaDiagramPanel

- تضبط إعدادات الرسم للحصول على رسومات عالية الجودة.
- تجمع الانتقالات المتعددة بين نفس الحالتين في تسمية واحدة للمساعدة في الوضوح.
- ترسم الحالات كدوائر وتضع أرقام الحالات بداخلها.
- ترسم الأسهم التي تمثل الانتقالات بين الحالات، مع تسميات توضح (رمز الإدخال، رمز الإزالة / رموز الإضافة إلى المكدس).
- تستخدم ألوانًا مختلفة للتمييز (مثل اللون الأحمر للانتقالات غير الحتمية إذا كان الوضع حتميًا).
- تضيف علامات خاصة للحالة الابتدائية (→) وحالات القبول (دائرة مزدوجة).(
- تتعامل مع الانتقالات الذاتية (self-loops) بشكل خاص (حلقات حول نفس الحالة).
- تتخلص من موارد الرسم بعد الانتهاء .أهميتها: تُحوّل التعريف المجرد للـ PDA إلى تمثيل بصري يساعد المستخدم على فهم بنية وسلوك الآلة.


محاكي النظرية الاحتسابية - الإصدار ٣,٠ (مع الرسوم البيانية)
آلات منتهية (FA)
آلات الدفع للأسفل (PDA)
آلة تورنغ (TM)
محدودة (Deterministic)
غير محدودة (Non-deterministic)
تعريف انتقالات الآلة (PDA)
@accept: q2
q0,a,Z;q0,AZ
q0,a,A;q0,AA
q0,b,A;q1,ε
q1,b,A;q1,ε
q1,ε,Z;q2,ε
الرسم البياني للحالات
q0
q1
q2
a,Z/AZ
a,A/AA
b,A/ε
ε,Z/ε
اختبار السلسلة
aabb
اختبر
سجل التتبع
(q0, aabb, Z)
-> δ(q0, a, Z) = (q0, AZ)
(q0, abb, ZA)
-> δ(q0, a, A) = (q0, AA)
(q0, bb, ZAA)
-> δ(q0, b, A) = (q1, ε)
(q1, b, ZA)
-> δ(q1, b, A) = (q1, ε)
(q1, ε, Z)
-> δ(q1, ε, Z) = (q2, ε)
(q2, ε, ε)
جدول الانتقالات
من
إدخال
إزالة
إلى
إضافة


محاكي النظرية الاحتسابية - الإصدار ٣,٠ (مع الرسوم البيانية)
آلات منتهية (FA)
آلات الدفع للأسفل (PDA)
آلة تورنغ (TM)
محدودة (Deterministic)
غير محدودة (Non-deterministic)
تعريف انتقالات الآلة (PDA)
@accept: q2
q0,a,Z;q0,AZ
q0,a,A;q0,AA
q0,b,A;q1,ε
q1,b,A;q1,ε
q1,ε,Z;q2,ε
بناء الآلة (PDA)
اختبار السلسلة
aab
اختبر
مرفوضة
سجل التتبع
الرسم البياني للحالات
جدول الانتقالات
من
إدخال
إزالة
إلى
إضافة
غير محددة

TU Machin

BuildTmButton_Click(object sender EventArgs e)

العمل: تُستدعى هذه الدالة عند النقر على زر بناء آلة تورينج. تقوم بـ:

- تحليل نص تعريف آلة تورينج المدخل بواسطة المستخدم باستخدام الدالة ParseTmلإنشاء كائن .currentTm
- تحديث جدول الانتقالات المرئي (tmTransitionTable) بناءً على الآلة الجديدة.
- تحديد جميع معرفات الحالات وإنشاء كائنات VisualStateلكل منها، مع تمييز حالات القبول والرفض.
- تحديد مواضع هذه الحالات بصريًا على لوحة الرسم.(tmDiagramPanel)
- طلب إعادة رسم لوحة الرسم لعرض الآلة.
- تحديث ملصق النتيجة (tmResultLabel) ليشير إلى أن الآلة جاهزة.
- تتضمن معالجة للأخطاء لعرض رسالة في حال فشل بناء الآلة .أهميتها: هي النقطة التي تبدأ عندها عملية تحويل تعريف آلة تورينج النصي إلى تمثيل داخلي وعرض رسومي في واجهة المستخدم.

RunTmSimulation(string input)

العمل: تبدأ هذه الدالة محاكاة آلة تورينج على سلسلة إدخال معينة بشكل غير متزامن. تقوم بـ:

- التحقق مما إذا كانت الآلة قد تم بناؤها.
- تهيئة شريط الآلة (tmTape) بسلسلة الإدخال وتعيين موضع الرأس والحالة الأولية.
- تحديث رسالة الحالة في الواجهة لتشير إلى أن المحاكاة قيد التشغيل.
- الدخول في حلقة محاكاة تستمر لعدد أقصى من الخطوات (١٠٠٠ خطوة).
- في كل خطوة :
  - التحقق مما إذا كانت الآلة قد وصلت إلى حالة قبول أو رفض.
  - قراءة الرمز تحت رأس الآلة.
  - البحث عن الانتقال المناسب في قاموس انتقالات الآلة.
  - تنفيذ الانتقال: كتابة رمز جديد على الشريط، تغيير الحالة، وتحريك الرأس.
  - تحديث شريط الآلة بصريًا.
  - إبطاء المحاكاة مؤقتًا (Task.Delay) للسماح بمراقبة التغييرات.
- إذا تجاوزت المحاكاة حد الخطوات دون الوصول إلى حالة قبول أو رفض، يتم الإشارة إلى ذلك .أهميتها: تنفذ المحاكاة الفعلية لآلة تورينج خطوة بخطوة وتحدث الواجهة لتعكس تقدم المحاكاة ونتائجها.

TmTapeVisualizer_Paint(object sender PaintEventArgs e)

العمل: هذه الدالة مسؤولة عن رسم الشريط (Tape) الحالي لآلة تورينج في لوحة .tmTapeVisualizerتقوم بـ:

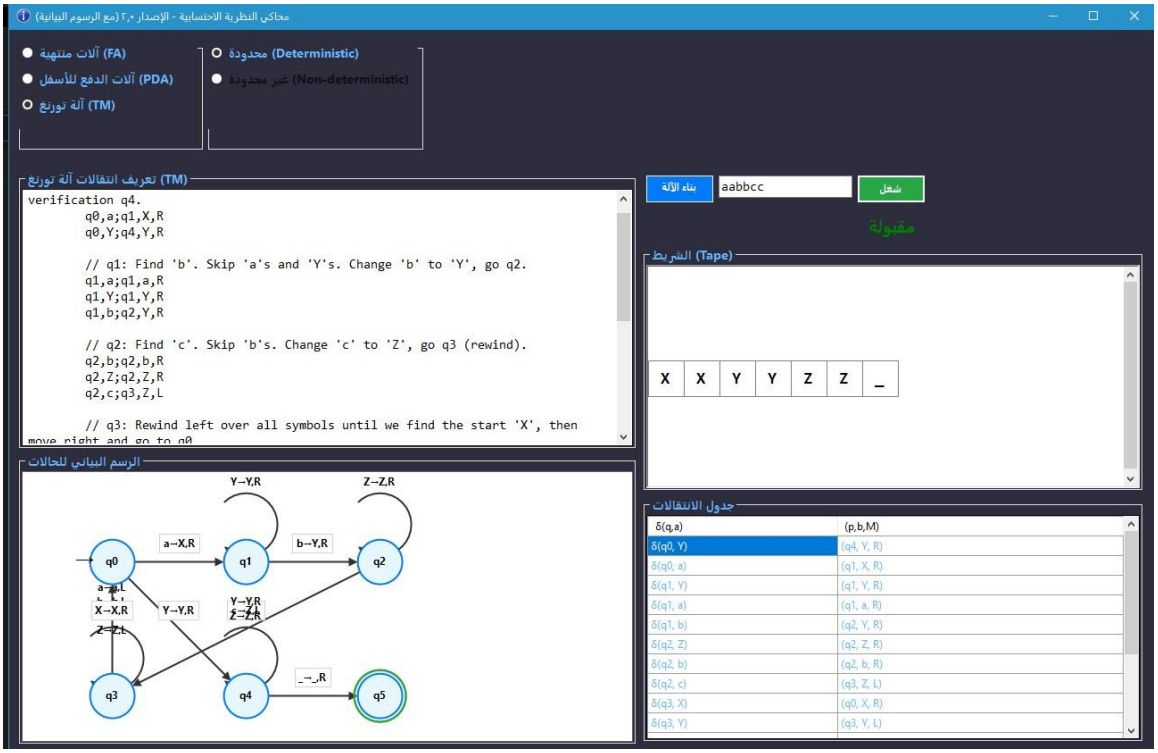
- مسح اللوحة.
- ضبط وضع التنعيم للرسومات.
- تحديد حجم الخلايا المرئية.

- حساب نطاق الخلايا التي يجب رسمها (من أول خلية مستخدمة إلى آخر خلية).
- رسم كل خلية كمستطيل.
- كتابة الرمز الموجود في كل خلية.
- تمييز الخلية التي يقع عندها رأس الآلة (الخلية النشطة) بلون خلفية مختلف ورسم مؤشر سهم فوقها.
- ضبط الحد الأدنى لحجم التمرير التلقائي للوحة لضمان ظهور كامل الشريط .أهميتها: توفر تمثيلاً بصريًا حيًا لتغيرات شريط آلة تورينج وموقع رأسها أثناء المحاكاة.

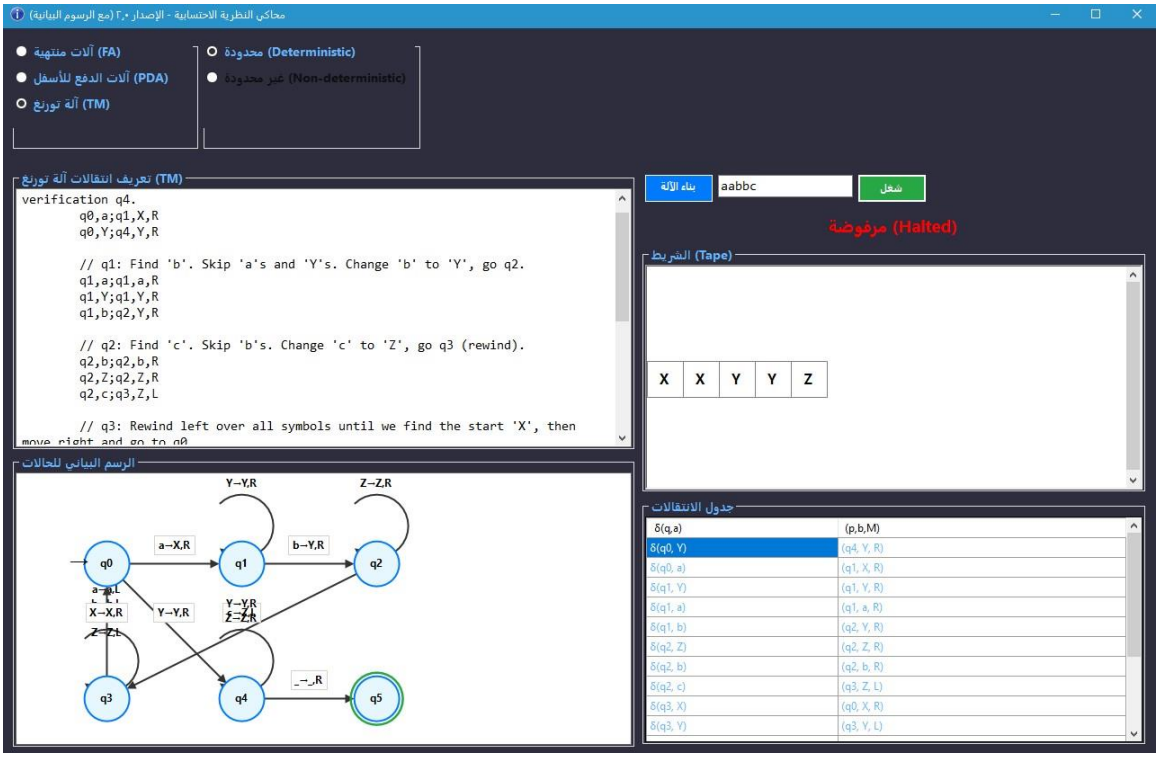
TmDiagramPanel_Paint(object sender PaintEventArgs e)

العمل: هذه الدالة مسؤولة عن رسم مخطط آلة تورينج على لوحة .tmDiagramPanelتقوم بـ:

- ضبط إزاحة التمرير التلقائي ووضع التنعيم.
- مسح اللوحة.
- حساب الحدود الفعلية للرسم لتحديد حجم التمرير التلقائي المطلوب.
- إعداد الفرش والأقلام المستخدمة للرسم (مثل ألوان الحالات، حالات القبول، الرفض، والانتقالات).
- تجميع الانتقالات التي تربط نفس الحالتين في تسمية واحدة لتبسيط الرسم.
- رسم جميع الانتقالات أولاً (تحت الحالات)، بما في ذلك الحلقات الذاتية (self-loops) والانتقالات بين حالات مختلفة، مع تسميات توضح الرمز المقروء/المكتوب واتجاه الحركة.
- رسم خلفية بيضاء لتسميات الانتقالات لتحسين الوضوح.
- رسم جميع الحالات كدوائر (فوق الانتقالات).
- تمييز حالات القبول بدائرة مزدوجة وحالات الرفض بدائرة حمراء.
- إضافة رمز ⟵للحالة الابتدائية.
- كتابة معرفات الحالات داخل الدوائر.
- تحرير موارد الرسم بعد الانتهاء .أهميتها: تُحوّل التعريف المجرد لآلة تورينج إلى تمثيل بصري يساعد المستخدم على فهم بنية وسلوك الآلة.



نلاحض ان السلسله المدخله للاختبار مقبولة لانها حققت شروط اله turing الذي تم دراستها سايقاً حيث ان عدد الـ a وعدد b وعدد c متساوياتان .



GetPointOnCircle(Point center Point target int radius)

العمل: هذه الدالة المساعدة تحسب نقطة على محيط دائرة تبدأ من centerباتجاه target، مع مسافة radiusمن المركز.

- تحسب الفرق في الإحداثيات (dx dy) بين النقطة المستهدفة والمركز.
- تحسب المسافة الإجمالية بين النقطتين.
- إذا كانت المسافة صفرًا، تُرجع المركز نفسه لتجنب القسمة على صفر.
- تحسب النسبة بين نصف القطر والمسافة.
- تستخدم هذه النسبة لتحديد إحداثيات النقطة الجديدة على محيط الدائرة .أهميتها: تُستخدم لضمان أن خطوط الانتقال في الرسم البياني للآلة تبدأ وتنتهي بدقة عند حافة دوائر الحالات، بدلاً من مركزها، مما يجعل الرسم أكثر دقة وجمالية.

AddConcatOperator(string pattern)

العمل: هذه الدالة تقوم بإضافة عامل الربط (.) الضمني إلى تعبير نمطي (Regular Expression) حيث يجب أن يحدث الربط.

تقوم بـ:

- المرور على كل حرف في النمط المدخل.
- إذا كان الحرف الحالي حرفًا أبجديًا رقميًا، أو قوسًا مغلقًا (أو علامة تكرار *، وكان الحرف التالي حرفًا أبجديًا رقميًا أو قوسًا مفتوحًا)يتم إضافة عامل الربط .بينهما .أهميتها: تُحوّل تعبيرات نمطية مثل abإلى a.b، أو a(b|c)إلى a.(b|c)، مما يجعلها متوافقة مع الخوارزميات التي تتطلب عوامل ربط صريحة.

InfixToPostfix(string p)

العمل: هذه الدالة تُحوّل تعبيرًا نمطيًا من صيغة التدوين التوسيطي (Infix Notation) إلى صيغة التدوين اللاحق Postfix) .(Notationتستخدم خوارزمية تكديس .(stack-based algorithm) تقوم بـ:

- تحديد أسبقية العوامل.(* . |)
- المرور على كل حرف في التعبير :
  - إذا كان حرفًا أبجديًا رقميًا، يُضاف مباشرة إلى التعبير اللاحق.
  - إذا كان قوسًا مفتوحًا)يُدفع إلى المكدس.
  - إذا كان قوسًا مغلقًا (يتم إزالة العوامل من المكدس وإضافتها إلى التعبير اللاحق حتى يتم العثور على قوس مفتوح مطابق.
  - إذا كان عاملًا (* . |) يتم إزالة العوامل ذات الأسبقية المتساوية أو الأعلى من المكدس وإضافتها إلى التعبير اللاحق قبل دفع العامل الحالي إلى المكدس.
- بعد معالجة جميع الأحرف، يتم إفراغ أي عوامل متبقية في المكدس إلى التعبير اللاحق.
- تتضمن معالجة للأخطاء للأقواس غير المتطابقة .أهميتها: تُسهّل عملية بناء الآلة ذات الحالات المنتهية غير الحتمية (NFA) من التعبير النمطي، حيث أن التدوين اللاحق يلغي الحاجة إلى الأقواس ويسهل معالجة العوامل.

PostfixToNfa(string postfix)

العمل: هذه الدالة تبني آلة ذات حالات منتهية غير حتمية (NFA) من تعبير نمطي بصيغة التدوين اللاحق (Postfix Notation) باستخدام خوارزمية تومبسون .(Thompson's Construction) تقوم بـ:

- استخدام مكدس لتخزين أجزاء الـ) NFA كائنات .(NfaFragment
- معالجة كل حرف في التعبير اللاحق :
  - إذا كان حرفًا أبجديًا رقميًا، تُنشئ NFA بسيطة بحالتين وانتقال واحد.
  - إذا كان عامل ربط (.) تُدمج آخر جزأين من NFA في المكدس لتشكيل تسلسل.
  - إذا كان عامل اختيار (|) تُدمج آخر جزأين من NFA لتشكيل اتحاد (اختيار.(
  - إذا كان عامل تكرار (*) تُطبق عملية كليين ستار على آخر NFA في المكدس.
- تُرجع قطعة الـ NFA النهائية بعد معالجة جميع الأحرف .أهميتها: تُحوّل التعبير النمطي اللاحق إلى بنيةNFA ، وهي خطوة أساسية في بناء الآلات الأوتوماتيكية من التعبيرات النمطية.

NfaToDfa(NfaFragment nfa IEnumerable&lt char> alphabet)

العمل: هذه الدالة تُحوّل آلة ذات حالات منتهية غير حتمية (NFA) إلى آلة ذات حالات منتهية حتمية (DFA) مكافئة باستخدام خوارزمية بناء المجموعة الجزئية .(Subset Construction) تقوم بـ:

- تهيئة قائمة انتظار للحالات غير المعلمة ومجموعة لـحالات الـ DFA التي تم إنشاؤها.
- حساب إغلاق إبسيلون للحالة الابتدائية للـ NFA لإنشاء الحالة الابتدائية للـ.DFA
- معالجة كل مجموعة حالات في قائمة الانتظار) تمثل حالة واحدة في الـ :(DFA
  - لكل رمز في الأبجدية، تحسب الدالة الانتقالات الممكنة من المجموعة الحالية ثم إغلاق إبسيلون لهذه الانتقالات.
  - إذا كانت المجموعة الناتجة جديدة، تُنشئ حالة DFA جديدة لها، وتضيفها إلى قائمة الانتظار ومجموعة حالات الـ.DFA

- تُضيف انتقالًا في الـ DFA بين الحالة الحالية والحالة الجديدة التي تمثل المجموعة الناتجة .أهميتها: تُنتج DFA مكافئة للـ NFA، وهي آلة أبسط وأكثر كفاءة للمحاكاة والتعرف على السلاسل.

EpsilonClosure(HashSet&lt State&gt states)

العمل: هذه الدالة تحسب "إغلاق إبسيلون" لمجموعة من حالات .NFA إغلاق إبسيلون لحالة هو مجموعة جميع الحالات التي يمكن الوصول إليها من الحالة الأصلية عن طريق اجتياز انتقالات إبسيلون (\0) فقط (بما في ذلك الحالة الأصلية نفسها). تقوم بـ:

- تهيئة مجموعة closureلتضم الحالات الأصلية.
- استخدام مكدس لتتبع الحالات التي يجب معالجتها.
- في حلقة، تقوم بإزالة حالة من المكدس :
  - إذا كان لهذه الحالة انتقالات إبسيلون، تُضاف الحالات التي يمكن الوصول إليها عبر هذه الانتقالات إلى closureوإلى المكدس إذا لم تكن موجودة بالفعل .أهميتها: تُستخدم في بناء الـ DFA من الـ NFA لمعالجة انتقالات إبسيلون، مما يضمن أن الـ DFA الناتجة تعترف بنفس اللغة التي تعترف بها الـNFA.

Move(HashSet&ltState&gt states char symbol)

العمل: تحسب هذه الدالة مجموعة الحالات التي يمكن الوصول إليها من مجموعة معينة من حالات NFA عند قراءة رمز إدخال معين. تقوم بـ:

- تهيئة مجموعة فارغة للنتائج.
- لكل حالة في المجموعة المدخلة :
  - إذا كانت الحالة تحتوي على انتقال للرمز المحدد، تُضاف جميع الحالات المستهدفة لهذا الانتقال إلى مجموعة النتائج .أهميتها: هي خطوة أساسية في خوارزمية بناء المجموعة الجزئية المستخدمة لتحويل NFA إلىDFA ، حيث تحدد الحالات التي يمكن الوصول إليها بعد استهلاك رمز معين.

SetToKey(HashSet&lt State&gt set)

العمل: هذه الدالة تحوّل مجموعة من كائنات Stateإلى مفتاح سلسلة فريد. تقوم بـ:

- أخذ معرفات (Id) جميع الحالات في المجموعة.
- ترتيب هذه المعرفات تصاعديًا.
- دمج المعرفات المرتبة في سلسلة واحدة، محاطة بأقواس متعرجة ومفصولة بفواصل (مثال .({5 3 1} :أهميتها: تُستخدم لإنشاء مفاتيح فريدة لتمثيل مجموعات حالات NFA عند تخزينها في قاموس لحالاتDFA ، مما يضمن أن كل مجموعة فريدة من حالات NFA تتوافق مع حالة DFA واحدة.

UpdateFaTransitionTable(State dfaStart List&lt State&gt allStates IEnumerable&lt char&gt alphabet)

العمل: هذه الدالة تقوم بتحديث جدول الانتقالات المرئي لآلة الحالات المنتهية (FA) في واجهة المستخدم. تقوم بـ:

- مسح الصفوف والأعمدة الحالية في الجدول.
- إضافة عمود للحالة وعمود لكل رمز في الأبجدية.
- لكل حالة في قائمة :allStates
  - إنشاء صف جديد.
  - إضافة اسم الحالة q) متبوعًا بالمعرف)، مع تمييز الحالة الابتدائية (→) وحالات القبول.(*)
  - لكل رمز في الأبجدية، إضافة الحالة التي يمكن الوصول إليها من الحالة الحالية عند قراءة الرمز (أو —إذا لم يكن هناك انتقال .(أهميتها: تعرض جدولًا واضحًا للانتقالات في الـDFA ، مما يساعد المستخدم على فهم كيفية عمل الآلة.

PositionStates(List&lt State> allStates Panel diagramPanel)

العمل: تقوم هذه الدالة بتحديد مواضع الحالات في الرسم البياني للآلة (FA) في شكل دائري على لوحة .diagramPanelتقوم بـ:

- حساب نصف قطر الدائرة التي ستوضع عليها الحالات، وتحديد مركز اللوحة.
- تقسيم محيط الدائرة بالتساوي بناءً على عدد الحالات.
- تعيين موضع) إحداثيات X و (Y لكل حالة بحيث توزع بالتساوي حول الدائرة .أهميتها: تُساعد في ترتيب الحالات بصريًا بطريقة منظمة على لوحة الرسم لسهولة القراءة والفهم.

) PositionStates(List&lt VisualState> states Panel panel)إصدار آخر(

العمل: هذا الإصدار من الدالة PositionStatesيوزع الحالات في صفوف وأعمدة على لوحة .panelتقوم بـ:

- تحديد نقطة بداية للموضع (هامش من اليسار والأعلى).
- تحديد مسافة ثابتة بين الحالات.
- حساب عدد الأعمدة والصفوف بناءً على العدد الإجمالي للحالات.
- تعيين موضع) إحداثيات X و (Y لكل حالة بناءً على موقعها في الشبكة (صف وعمود) .أهميتها: توفر طريقة بديلة لتوزيع الحالات بصريًا، وهي مفيدة بشكل خاص عندما يكون هناك عدد كبير من الحالات التي لا تتناسب مع الترتيب الدائري

SimulateNPD(PushdownAutomaton pda string input bool isDeterministicMode)

العمل: هذه الدالة تقوم بمحاكاة آلة الدفع للأسفل (PDA) على سلسلة إدخال معينة. تستخدم خوارزمية البحث في العرض (BFS) لاستكشاف جميع المسارات الممكنة في الـ PDA (خاصة للآلات غير الحتمية). تقوم بـ:

- تهيئة قائمة انتظار (Queue) تحتوي على التكوين الأولي للآلة (الحالة الابتدائية، مؤشر الإدخال، المكدس الأولي، وسجل التتبع.(
- في حلقة، تستمر المحاكاة حتى تفرغ قائمة الانتظار أو يتم تجاوز حد معين من الخطوات (٢٠٠٠ خطوة).

- في كل خطوة :
  - إزالة تكوين من قائمة الانتظار.
  - تسجيل التكوين الحالي في سجل التتبع.
  - التحقق مما إذا كانت السلسلة مقبولة (نهاية الإدخال وحالة قبول).
  - تحديد جميع الانتقالات الممكنة من الحالة الحالية بناءً على رمز الإدخال الحالي ورمز أعلى المكدس.
  - إذا كان isDeterministicModeصحيحًا ووجد أكثر من انتقال ممكن، تُعلن انتهاك الحتمية.
  - لكل انتقال ممكن، يتم إنشاء تكوين جديد (حالة جديدة، مؤشر إدخال محدث، مكدس محدث) وإضافته إلى قائمة الانتظار.
- تُرجع نتيجة المحاكاة (مقبولة/مرفوضة)، سجل التتبع، وما إذا كان هناك انتهاك للحتمية .أهميتها: هي المحرك الفعلي الذي يقوم بتشغيل الـ PDA على سلسلة الإدخال، مع القدرة على التعامل مع الطبيعة الحتمية وغير الحتمية للآلة وتسجيل مسار المحاكاة.

ParseTm(string definition)

العمل: هذه الدالة مسؤولة عن تحليل تعريف آلة تورينج النصي الذي يُدخله المستخدم وتحويله إلى كائن .TuringMachineتقوم بـ:

- تقسيم سلسلة التعريف إلى سطور.
- استخدام تعبير نمطي (Regex) لمطابقة صيغة كل سطر انتقال (مثال:(q0 a q1 b R
- تحليل سطور التعريف الخاصة بحالة البداية (START_STATE:) حالات القبول (ACCEPT_STATES:) وحالة الرفض (REJECT_STATE:).
- تحليل الحالة المصدر، رمز القراءة، الحالة الوجهة، رمز الكتابة، واتجاه الحركة (يمين R، يسار L، أو ثبات .(S
- تجميع جميع معرفات الحالات الموجودة.
- إضافة الانتقالات إلى قاموس Transitionsالخاص بكائن .TuringMachine
- تعيين حالات البداية والقبول والرفض، مع استخدام قيم افتراضية إذا لم يتم تحديدها صراحة في التعريف.
- تتضمن معالجة الأخطاء للتحقق من صحة صيغة الإدخال أو عدم وجود حالات معرفة .أهميتها: هي الدالة الأساسية التي تُحوّل التعريف النصي لآلة تورينج إلى كائن برمجي يمكن محاكاته، وتتحقق من صحة بنية التعريف.

UpdateTmTransitionTable(TuringMachine tm)

العمل: هذه الدالة تقوم بتحديث جدول الانتقالات المرئي لآلة تورينج في واجهة المستخدم. تقوم بـ:

- مسح الصفوف والأعمدة الحالية في الجدول.
- إضافة عمودين: الأول لـ) (q a)δالانتقال من الحالة qبقراءة الرمز (a، والثاني لـ) (p b M)الانتقال إلى الحالة pبكتابة الرمز b والتحرك .(M
- المرور على جميع الانتقالات في كائن TuringMachine، وترتيبها حسب الحالة ورمز القراءة.
- إضافة كل انتقال كصف جديد إلى الجدول، مع تنسيق مناسب للمعلومات.
- ضبط وضعية حجم الأعمدة لملء المساحة المتاحة .أهميتها: توفر تمثيلاً جدوليًا واضحًا لجميع انتقالات آلة تورينج، مما يساعد المستخدم على مراجعة وفهم سلوك الآلة.