UNIVERSITATEA TEHNICĂ „GHEORGHE ASACHI” IAȘI
FACULTATEA DE AUTOMATICĂ ȘI CALCULATOARE
DOMENIUL: Calculatoare și Tehnologia Informației
SPECIALIZAREA: Calculatoare

Inteligență Artificială - Proiect

Inferența predicativă prin raționament înainte

Coordonator,
Asist. drd. ing. Mircea Hulea

Autori,
Gîscă Valentin, 1405B
Lupu Andreea-Sabina, 1405B

IAȘI

2024-2025

Cuprins

1. Descrierea problemei considerate

Problema abordată în acest proiect se referă la dezvoltarea unui sistem de inferență predicativă utilizând raționamentul înainte. Obiectivul principal este de a demonstra cum pot fi aplicate reguli logice pentru a deriva concluzii pornind de la o bază de fapte și reguli predefinite.

Proiectul consideră exemple matematice relevante care implică premise simple și explicabile. Sistemul trebuie să fie capabil să proceseze regulile furnizate, să interpreteze faptele și să determine concluziile valabile.

2. Aspecte teoretice privind algoritmul

2.1 Raționamentul înainte

Raționamentul înainte este un proces logic care pleacă de la o serie de fapte inițiale și aplică reguli pentru a genera noi fapte până la atingerea unui obiectiv sau până când nu mai pot fi generate fapte noi. Se bazează pe următoarele componente principale:

- Baza de fapte: un set de propoziții care sunt considerate adevărate la un moment dat.
- Baza de reguli: un set de reguli logice de forma "Dacă (premise) atunci (concluzie)".
- Motorul de inferență: algoritmul care aplică regulile asupra faptelor pentru a deduce noi cunoștințe.

2.2 Teorema lui Pitagora

Teorema lui Pitagora este un rezultat fundamental în geometria euclidiană care afirmă că într-un triunghi dreptunghic, pătratul lungimii ipotenuzei este egal cu suma pătratelor lungimilor catetelor. Formula matematică este:

$$a^2 + b^2 = c^2$$

unde:

- a și b - catetele triunghiului dreptunghic;
- c - ipotenuza.

Această teoremă este utilizată în proiect pentru a deduce lungimea unei cateta sau a unei ipotenuze când celelalte două laturi sunt cunoscute.

2.3 Principiul Includerii-Excluderii

Principiul Includerii-Excluderii este utilizat în teoria mulțimilor pentru a calcula cardinalitatea uniunii mai multor mulțimi. Formula pentru două mulțimi A și B este:

$$|A \cup B| = |A| + |B| - |A \cap B|$$

Această formulă se extinde pentru mai multe mulțimi, ajustând termenii de intersecție pentru a evita numărarea multiplă. În proiect, principiul este utilizat pentru a calcula corect cardinalitatea uniunii mulțimilor pornind de la datele introduse de utilizator.

2.4 Avantaje

- Permite explorarea exhaustivă a spațiului de soluții.
- Simplu de implementat pentru probleme bine definite cu reguli clare.

2.5 Limite

- Ineficient pentru baze de reguli foarte mari.
- Posibilitatea generării de multe fapte irelevante dacă regulile nu sunt bine definite.

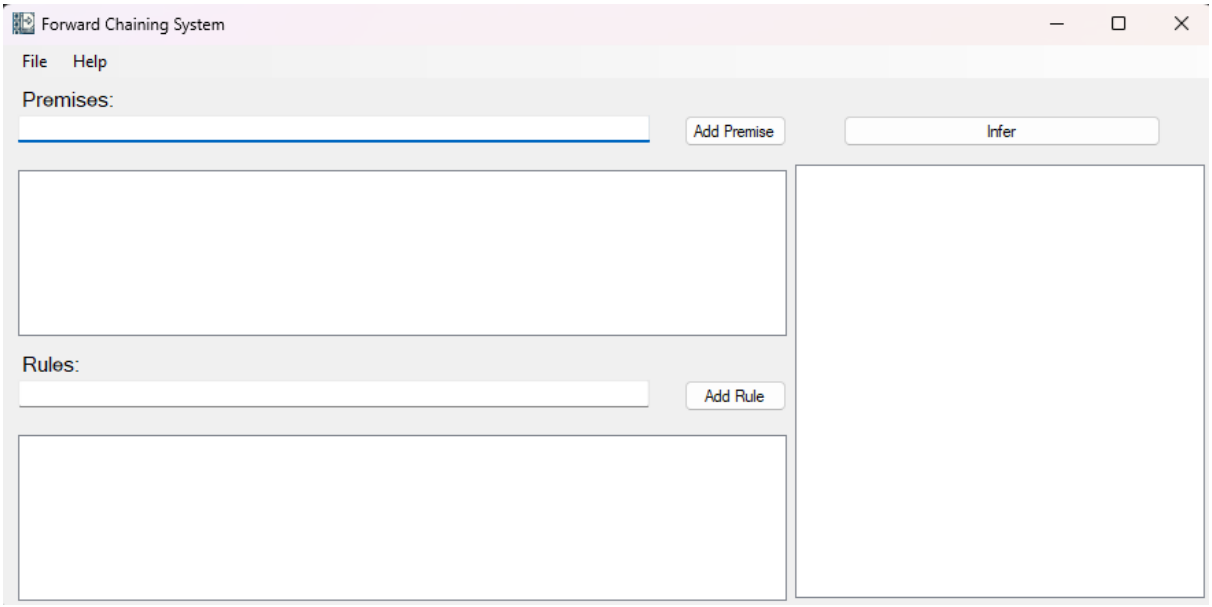
3. Modalitatea de rezolvare

Proiectul implementează un algoritm de inferență predicativă prin raționament înainte în care utilizatorul se folosește de interfață pentru a obține rezultatele căutate.

În continuare se va descrie interfața pentru a se înțelege modalitatea de lucru.

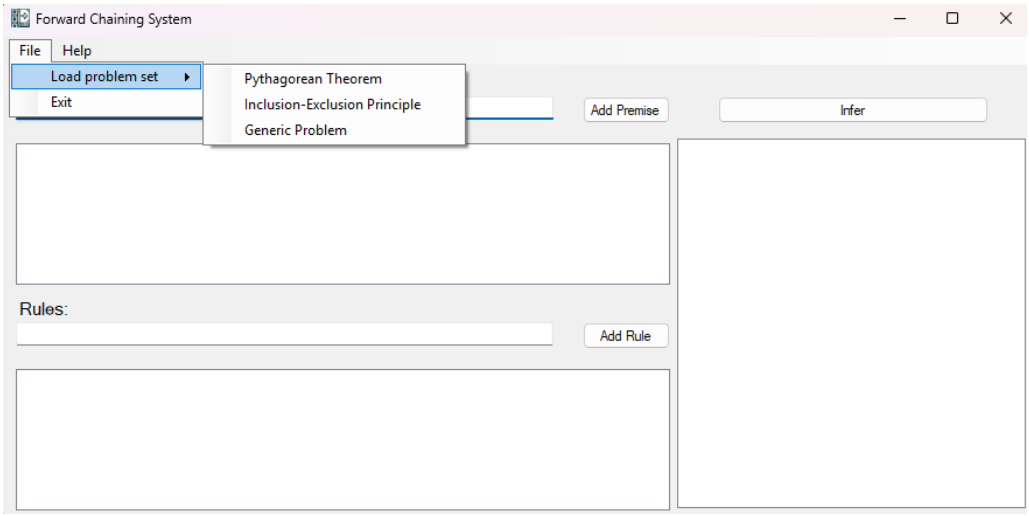
3.1 Interfața utilizator

Interfața principală a aplicației permite utilizatorului să interacționeze cu sistemul. Aceasta include meniuri pentru gestionarea metodelor, introducerea premiselor și selectarea principiilor.



3.2 Meniul de încărcare a metodelor predefinite

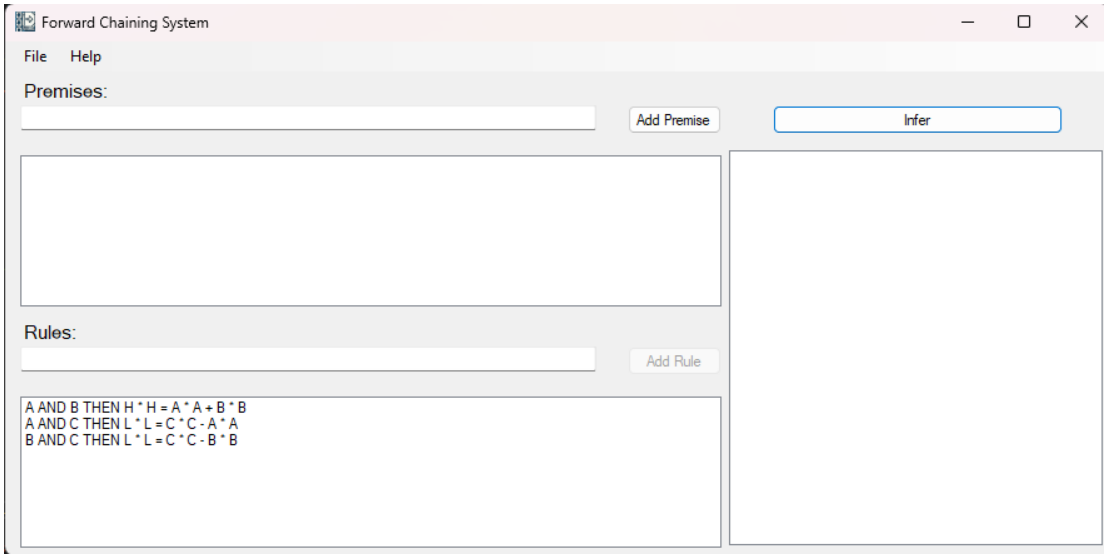
Utilizatorii pot alege din meniul de încărcare metoda predefinită dorită, apoi completa cu premisele și regulile dorite.



3.3 Activarea principiilor matematice

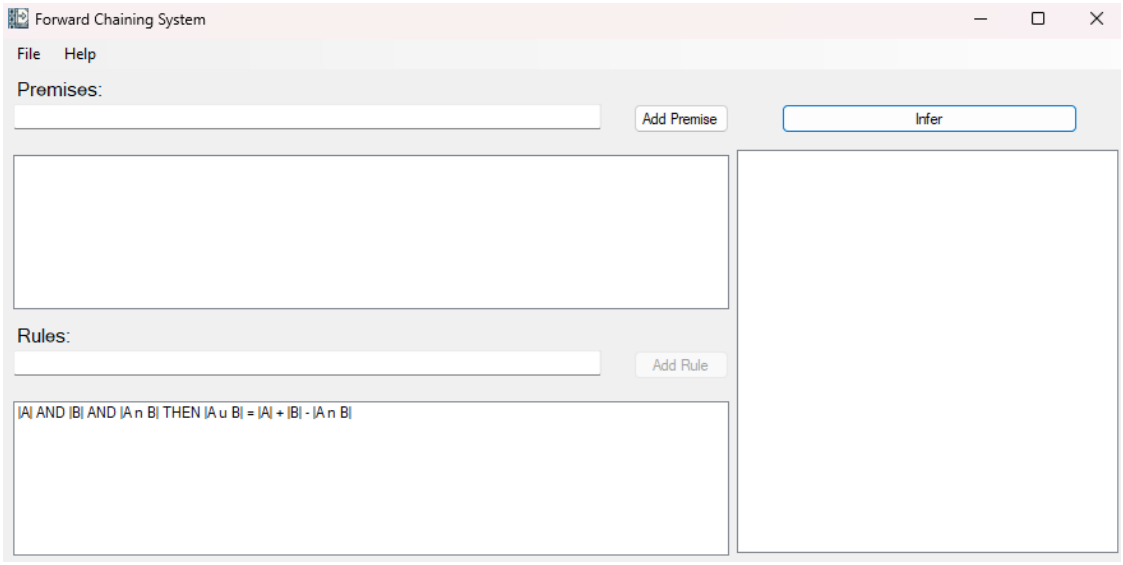
3.3.1 Teorema lui Pitagora

Pentru a activa logica teoremei lui Pitagora, utilizatorul trebuie să selecteze principiul și să introducă valori numerice (în Premises) dorite.



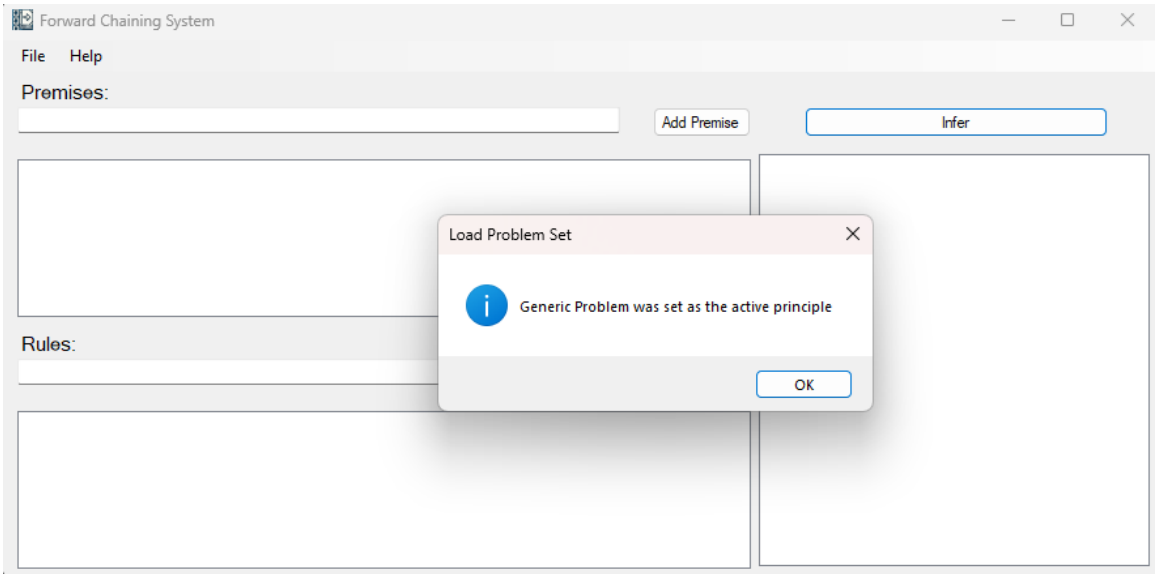
3.3.2 Principiul Includerii-Excluderii

Activarea acestui principiu presupune selectarea logicii corespunzătoare și introducerea dimensiunilor mulțimilor și intersecțiilor lor.



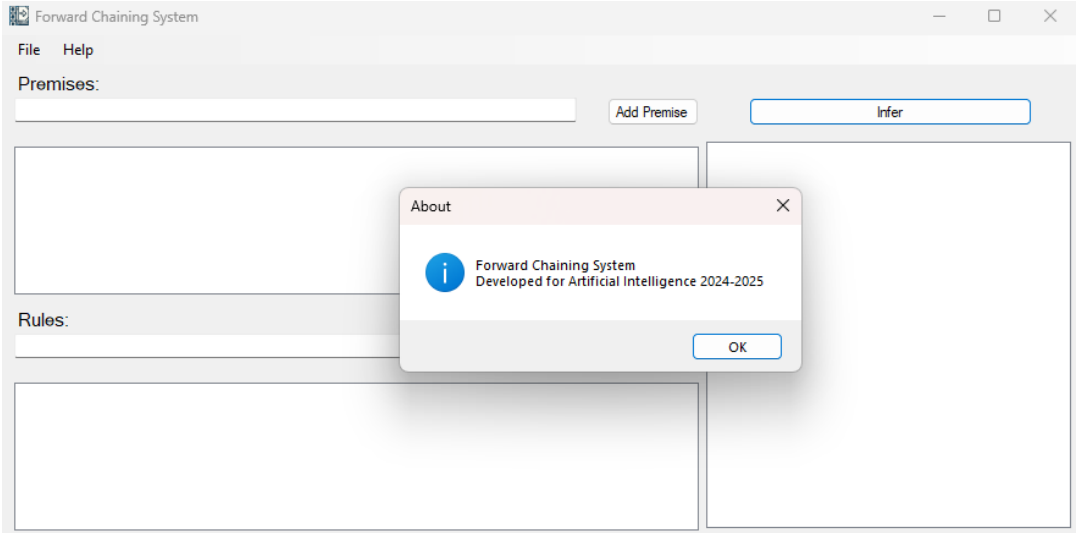
3.4 Configurarea generică

Sistemul permite selectarea unor setări generice active pentru testarea altor reguli sau premise.



3.5 Fereastra ”Despre aplicație”

Pentru mai multe informații, utilizatorii pot accesa secțiunea "Despre aplicație".



Aceste componente ilustrează modul în care interfața grafică facilitează utilizarea sistemului de inferență.

4. Cod sursă, explicații, comentarii - blocuri semnificative

4.1. MainForm.cs

Acest fișier gestionează interfața grafică a aplicației și permite utilizatorului să introducă premise și reguli.

- **Metoda MainForm:** Creează o bază de cunoștințe pentru stocarea faptelor și regulilor.

```
1 reference
public MainForm()
{
    InitializeComponent();
    mKnowledgeBase = new KnowledgeBase();
    mKnowledgeBase.SetActivePrinciple(new GenericPrinciple());
}
```

- **Adăugarea de premise:** Permite utilizatorului să introducă premise în sistem.

```
private void premisesButton_Click(object sender, EventArgs e)
{
    string premise = premisesBox.Text.Trim();
    if (!string.IsNullOrEmpty(premise))
    {
        try
        {
            mKnowledgeBase.AddFact(premise);
            premisesListBox.Items.Add(premise);
            premisesBox.Clear();
        }
        catch (Exception ex)
        {
            MessageBox.Show($"Error adding premise: {ex.Message}", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);
        }
    }
}
```

- **Inferență și afișare rezultate:** Aplică regulile asupra premiselor și afișează rezultatele.

```
1 reference
private void inferButton_Click(object sender, EventArgs e)
{
    try
    {
        List<string> newFacts = mKnowledgeBase.Infer();
        conclusionsList.Items.Clear();
        foreach (var fact in newFacts)
        {
            conclusionsList.Items.Add(fact);
        }

        if (newFacts.Count == 0)
        {
            MessageBox.Show("No new conclusions could be inferred.", "Inference Complete", MessageBoxButtons.OK, MessageBoxIcon.Information);
        }
    }
    catch (Exception ex)
    {
        MessageBox.Show($"Error during inference: {ex.Message}", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
}
```

- **Metoda inclusionExclusionPrincipleToolStripMenuItem_Click:** Această metodă setează Principiul Includerii-Excluderii ca principiu activ și adaugă regula de calcul al uniunii mulțimilor folosind principiul respectivă.

```
1 reference
private void inclusionExclusionPrincipleToolStripMenuItem_Click(object sender, EventArgs e)
{
    MessageBox.Show("Inclusion-Exclusion Principle was set as the active principle!", "Load Problem Set", MessageBoxButtons.OK, MessageBoxIcon.Information);

    rulesBox.Clear();
    rulesListBox.Items.Clear();
    premisesBox.Clear();
    premisesListBox.Items.Clear();
    conclusionsList.Items.Clear();
    ruleButton.Enabled = false;

    mKnowledgeBase.SetActivePrinciple(new InclusionExclusionPrinciple());

    mKnowledgeBase.AddRule("|A| AND |B| AND |A n B| THEN |A u B| = |A| + |B| - |A n B|");

    rulesListBox.Items.Add("|A| AND |B| AND |A n B| THEN |A u B| = |A| + |B| - |A n B|");
}
```

- **Metoda pythagoreanTheoremToolStripMenuItem_Click:** Această metodă setează Teorema lui Pitagora ca principiu activ și adaugă reguli de calcul unei laturi necunoscute folosind teorema respectivă.

```
1 reference
private void pythagoreanTheoremToolStripMenuItem_Click(object sender, EventArgs e)
{
    MessageBox.Show("Pythagorean Theorem was set as the active principle!", "Load Problem Set", MessageBoxButtons.OK, MessageBoxIcon.Information);

    rulesBox.Clear();
    rulesListBox.Items.Clear();
    premisesBox.Clear();
    premisesListBox.Items.Clear();
    conclusionsList.Items.Clear();
    ruleButton.Enabled = false;

    mKnowledgeBase.SetActivePrinciple(new PythagoreanPrinciple());

    mKnowledgeBase.AddRule("A AND B THEN H * H = A * A + B * B");
    mKnowledgeBase.AddRule("A AND C THEN L * L = C * C - A * A");
    mKnowledgeBase.AddRule("B AND C THEN L * L = C * C - B * B");

    rulesListBox.Items.Add("A AND B THEN H * H = A * A + B * B");
    rulesListBox.Items.Add("A AND C THEN L * L = C * C - A * A");
    rulesListBox.Items.Add("B AND C THEN L * L = C * C - B * B");

}
```

4.2 PythagoreanPrinciple.cs

Această clasă implementează logica necesară pentru demonstrarea Teoremei lui Pitagora.

- **Constructorul:** Inițializează baza de fapte numerice și regulile asociate principiului.

```
1 reference
public PythagoreanPrinciple()
{
    numericalFacts = new Dictionary<string, double>();
    rules = new List<Rule>();
}
```

- **Adăugarea de premise:** Permite introducerea valorilor numerice necesare rezolvării.

```
2 references
public void AddPremise(string premise)
{
    var parts = premise.Split('=');
    if (parts.Length == 2)
    {
        string key = parts[0].Trim();
        if (double.TryParse(parts[1].Trim(), out double value))
        {
            numericalFacts[key] = value;
        }
    }
}
```

- **Reguli și inferență:** Deduce latura necunoscută a triunghiului.

```
2 references
public List<string> Infer()
{
    var results = new List<string>();

    foreach (var rule in rules)
    {
        if (rule.IsSatisfied(numericalFacts.Keys.ToHashSet()))
        {
            string result = rule.EvaluateConclusion(numericalFacts, this);
            results.Add($"{rule.Conclusion.Split('*')[0].Trim()} = {result}");
        }
    }

    return results;
}
```

4.3 InclusionExclusionPrinciple.cs

Această clasă implementează principiul includerii-excluderii.

- **Constructorul:** Crează structura pentru stocarea faptelor numerice și regulilor.

```
1 reference
public InclusionExclusionPrinciple()
{
    numericalFacts = new Dictionary<string, double>();
    rules = new List<Rule>();
}
```

- **Adăugarea de premise:** Permite introducerea mărimilor mulțimilor și intersecțiilor lor.

```
2 references
public void AddPremise(string premise)
{
    var parts = premise.Split('=');
    if (parts.Length == 2)
    {
        string key = parts[0].Replace("|", "").Trim();
        if (double.TryParse(parts[1].Trim(), out double value))
        {
            numericalFacts[key] = value;
        }
    }
}
```

- **Reguli și inferență:** Aplică principiul pentru a calcula uniunile mulțimilor.

```
2 references
public List<string> Infer()
{
    var results = new List<string>();

    foreach (var rule in rules)
    {
        if (rule.IsSatisfied(numericalFacts.Keys.ToHashSet()))
        {
            string result = rule.EvaluateConclusion(numericalFacts, this);
            results.Add($"{rule.Conclusion.Split('=')[0].Trim()} = {result}");
        }
    }

    return results;
}
}
```

4.4 Rule.cs

Această clasă gestionează structura regulilor utilizate în inferență.

O metodă importantă din acest fișier este **EvaluateConclusion** deoarece aceasta aplică regula pentru a calcula rezultatul în funcție de principiul activ.

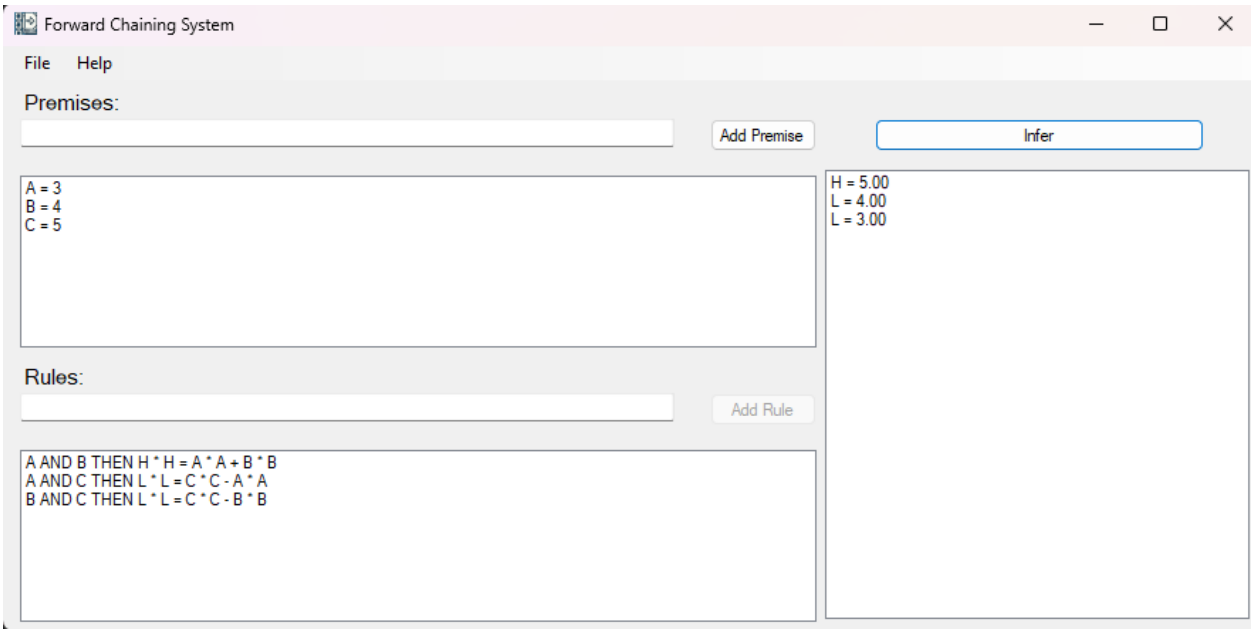
```
3 references
public string EvaluateConclusion(Dictionary<string, double> numericalFacts, IPrinciple activePrinciple)
{
    string processedConclusion = Conclusion;

    if(activePrinciple.Name == "Pythagorean")
    {
        string[] equationParts = Conclusion.Split('=');
        if (equationParts.Length != 2)
            throw new Exception("Invalid conclusion format.");

        string leftHandSide = equationParts[0].Trim();
        string rightHandSide = equationParts[1].Trim();

        bool isAddition = rightHandSide.Contains("+");
        bool isSubtraction = rightHandSide.Contains("-");
        if (!isAddition && !isSubtraction)
            throw new Exception("The equation must contain either '+' or '-' for Pythagorean-like calculations.");

        string unknownVariable = leftHandSide.Split('*')[0].Trim();

        string[] terms = rightHandSide.Split(new[] { '+', '-' }, StringSplitOptions.RemoveEmptyEntries);
        if (terms.Length != 2)
            throw new Exception("The equation must involve two terms for Pythagorean-like calculations.");

        string term1 = terms[0].Split('*')[0].Trim();
        string term2 = terms[1].Split('*')[0].Trim();

        if (!numericalFacts.ContainsKey(term1) || !numericalFacts.ContainsKey(term2))
            throw new Exception("Missing required numerical facts for calculation.");

        double value1 = numericalFacts[term1];
        double value2 = numericalFacts[term2];

        double result;
        if (isAddition)
        {
            result = Math.Sqrt(value1 * value1 + value2 * value2);
        }
        else
        {
            if (value1 * value1 - value2 * value2 < 0)
                throw new Exception("Invalid values: cannot calculate square root of a negative number.");
            result = Math.Sqrt(value1 * value1 - value2 * value2);
        }
        return $"{result:F2}";
    }
    else if(activePrinciple.Name == "Inclusion-Exclusion")
    {
        foreach (var fact in numericalFacts)
        {
            processedConclusion = processedConclusion.Replace($"|{fact.Key}|", fact.Value.ToString());
        }
        using (var table = new System.Data.DataTable())
        {
            return table.Compute(processedConclusion.Split('=')[1], "").ToString();
        }
    }
    // Generic principle
    else
    {
        try
        {
            foreach (var fact in numericalFacts)
            {
                processedConclusion = processedConclusion.Replace(fact.Key, fact.Value.ToString());
            }

            // Evaluate the processed conclusion using DataTable
            using (var table = new System.Data.DataTable())
            {
                string result = table.Compute(processedConclusion.Split('=')[1], "").ToString();
                return $"{processedConclusion.Split('=')[0].Trim()} = {result}";
            }
        }
        catch (Exception ex)
        {
            return $"Error evaluating generic principle: {ex.Message}";
        }
    }
}
```

5. Rezultate obținute

Rezultatele obținute demonstrează funcționalitatea sistemului de inferență prin raționament înainte. Acestea includ deducerea corectă a faptelor necunoscute pe baza premiselor introduse de utilizator și aplicarea regulilor specifice celor două principii matematice implementate: Teorema lui Pitagora și Principiul Includerii-Excluderii.

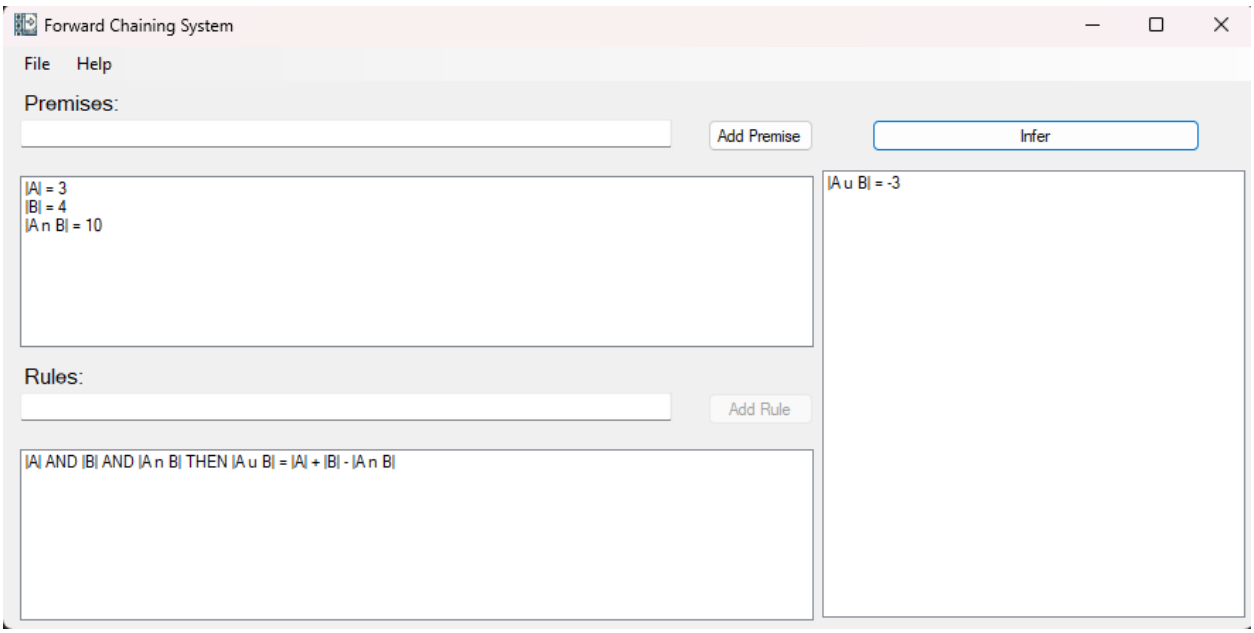
Teorema lui Pitagora

- **Date introduse:** A=3 B=4 C=5
- **Reguli aplicate:** A AND B THEN H*H=A*A+B*B (aflare ipotenuză)
  A AND C THEN L*L=C*C-A*A (aflare catetă)
  B AND C THEN L*L=C*C-B*B (aflare catetă)
- **Rezultate obținut pentru fiecare regulă:** H=5.00
  L=4.00
  L=3.00



Principiul Includerii-Excluderii

- **Date introduse:** |A|=3 |B|=4 |A∩B|=10
- **Regulă aplicată:** |A| AND |B| AND |A∩B| THEN |A U B| = |A|+|B|-|A∩B|
- **Rezultat obținut:** |A U B| = 3



6. Concluzii

Proiectul demonstrează aplicabilitatea algoritmului de inferență predicativă pentru rezolvarea problemelor matematice. Integrarea Teoremei lui Pitagora și a Principiului Includerii-Excluderii a evidențiat flexibilitatea și eficiența motorului de inferență în derivarea concluziilor corecte.

Sistemul oferă o interfață intuitivă care facilitează utilizarea și permite testarea diverselor scenarii. Această abordare poate fi extinsă și pentru alte principii matematice complexe sau pentru domenii diverse.

7. Rolul membrilor din echipă

- **Gîscă Valentin** – crearea interfețelor generice + implementarea metodei pentru Principiul Includerii-Excluderii
- **Lupu Andreea-Sabina** – implementare metodei pentru Teorema lui Pitagora + documentație proiect

Codul sursă este încărcat pe github, unde se pot vizualiza și commit-urile în funcție de sarcinile atribuite.

Link: https://github.com/Trased/ForwardChainingSystem

8. Bibliografie

- "Artificial Intelligence: A Modern Approach" - Stuart Russell, Peter Norvig.
- https://brilliant.org/wiki/principle-of-inclusion-and-exclusion-pie/
- https://brilliant.org/wiki/pythagorean-theorem/
- https://builtin.com/artificial-intelligence/forward-chaining-vs-backward-chaining