Aplicație de Organizare a Orarului folosind Forward Checking

Olaru Ciprian-Timotei, grupa 1410A

1. Descrierea problemei

Proiectul propune realizarea unei aplicații software pentru organizarea unui orar, utilizând algoritmul **Forward Checking**. Scopul aplicației este de a aloca grupe studenților în săli disponibile, respectând o serie de constrângeri specifice, cum ar fi:

- **Disponibilitatea sălilor**: fiecare sală are intervale orare care pot fi ocupate.
- **Disponibilitatea grupelor**: fiecare grupă are propriile intervale orare preferate.
- Evitarea suprapunerilor: o sală nu poate fi alocată la mai multe grupe în același interval orar.
- Condiții de unicitate: fiecare grupă are un nume unic.

Aplicația permite adăugarea manuală de săli și grupe, generarea automată a orarului pe baza constrângerilor, rezolvarea problemelor prin utilizarea algoritmului de **Forward Checking**, vizualizarea și gestionarea orarului generat și resetarea orarului și ștergerea rezervărilor din săli.

2. Aspecte teoretice

Forward Checking este un algoritm utilizat în rezolvarea problemelor de tip CSP (Constraint Satisfaction Problems).

Caracteristici principale ale algoritmului:

- 1. **Identificarea soluțiilor valide** algoritmul verifică posibilitatea alocării unei valori (interval) pentru o variabilă (grupă) fără a încălca constrângerile impuse de alte variabile.
- 2. **Pruning (tăierea domeniilor invalide)** după fiecare alocare, elimină din domeniile celorlalte variabile valorile care nu mai sunt posibile.
- 3. **Backtracking controlat** algoritmul revine la pasul anterior doar dacă nu mai există soluții valide pentru variabila curentă.
- 4. **Eficiență** comparativ cu alte metode (ex. Backtracking), Forward Checking reduce spațiul de căutare, fiind mai rapid în cazul problemelor complexe cu multe constrângeri.

3. Modalitatea de rezolvare

Pentru realizarea aplicației, am folosit algoritmul Forward Checking în următorii pași:

1. Structurarea datelor:

- Sălile sunt reprezentate printr-o clasă Sala, care include numele sălii și intervalele orare rezervate.
- Grupele sunt reprezentate ca tuple (string grupa, List<(int Start, int End)> intervale).
- Orarul generat este o listă de tuple (string Grupa, string Sala, (int Start, int End) Interval).

2. Implementarea funcțiilor principale:

- Adăugarea de săli și grupe: utilizatorii pot introduce manual datele.
- **Generarea orarului**: funcția RezolvaOrar alocă grupe în săli folosind Forward Checking.
- **Validarea**: funcția EsteValid verifică dacă o sală poate fi alocată pentru o anumită grupă într-un interval dat.

3. Interfața grafică:

- Aplicația utilizează Windows Forms pentru afișarea datelor și interacțiunea cu utilizatorul.
- o DataGridView este folosit pentru vizualizarea orarului generat.

4. Resetare:

• Rezervările sunt șterse împreună cu orarul prin funcția btnStergeOrar_Click.

4. Listarea părților semnificative din codul sursă

Adăugarea grupelor:

```
c/c++
private void btnAdaugaGrupa_Click(object sender, EventArgs e)
{
    if (!string.IsNullOrWhiteSpace(txtGrupa.Text) &&
!string.IsNullOrWhiteSpace(txtIntervale.Text))
    {
        if (grupe.Any(g => g.grupa.Equals(txtGrupa.Text,
        StringComparison.OrdinalIgnoreCase)))
        {
            MessageBox.Show($"Grupa {txtGrupa.Text} exista deja", "Eroare",
        MessageBoxButtons.OK, MessageBoxIcon.Error);
            return;
        }
        var intervale = new List<(int Start, int End)>();
        foreach (var intervalString in txtIntervale.Text.Split(','))
        {
            var bounds = intervalString.Trim().Split('-');
            if (bounds.Length == 2 && int.TryParse(bounds[0], out int start) &&
int.TryParse(bounds[1], out int end))
```

```
{
    intervale.Add((start, end));
}

grupe.Add((txtGrupa.Text, intervale));
    listGrupe.Items.Add($"{txtGrupa.Text} (Intervale: {string.Join(", ", intervale.Select(i => $"({i.Start}, {i.End})"))})");
}
}
```

Generarea orarului cu Forward Checking:

```
C/C++
private bool RezolvaOrar(List<(string Grupa, string Sala, (int Start, int End)
Interval) orar, List<(string grupa, List<(int Start, int End) intervale) grupe,
List<Sala> sali)
  if (grupe.Count == 0) return true;
  var grupaCurenta = grupe[0];
  var grupeRamase = grupe.GetRange(1, grupe.Count - 1);
  foreach (var interval in grupaCurenta.intervale)
     foreach (var sala in sali)
       if (EsteValid(orar, grupaCurenta.grupa, sala, interval.Start, interval.End))
         orar.Add((grupaCurenta.grupa, sala.Nume, interval));
         RezervaSala(sala, interval.Start, interval.End);
         if (RezolvaOrar(orar, grupeRamase, sali))
            return true;
         orar.RemoveAll(o => o.Grupa == grupaCurenta.grupa && o.Sala ==
sala.Nume && o.Interval == interval);
         sala.IntervaleRezervate.RemoveAll(i => i.Start == interval.Start && i.End
== interval.End);
  return false;
```

Funcție de validare:

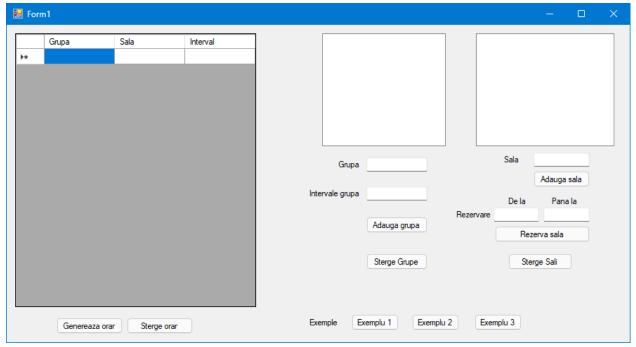
```
c/C++
private bool EsteValid(List<(string Grupa, string Sala, (int Start, int End) Interval)>
orar, string grupa, Sala sala, int intervalStart, int intervalEnd)
{
    // verificare interval
    foreach (var rezervare in sala.IntervaleRezervate)
    {
        if (rezervare.Start < intervalEnd && intervalStart < rezervare.End)
            return false;
    }

    // verificare grupa la un alt interval
    foreach (var o in orar)
    {
        if (o.Grupa == grupa && o.Interval.Start < intervalEnd && intervalStart <
        o.Interval.End)
            return false;
    }

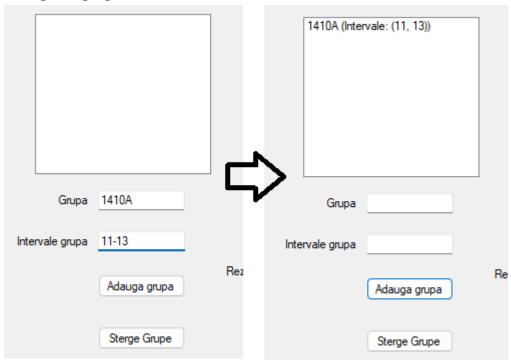
    return true;
}
```

5. Rezultatele obținute

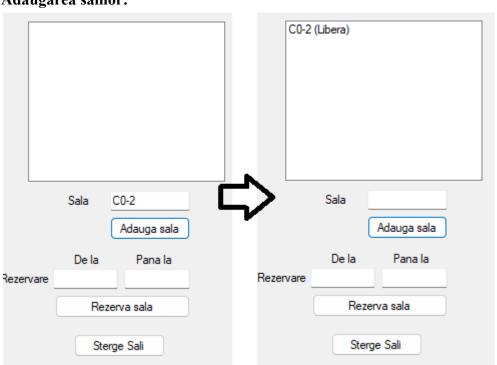
Interfața grafică a aplicației:



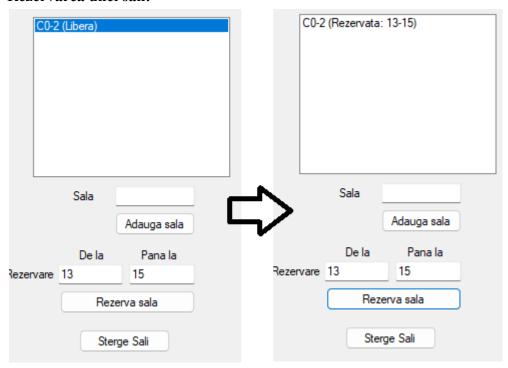
Adăugarea grupelor:



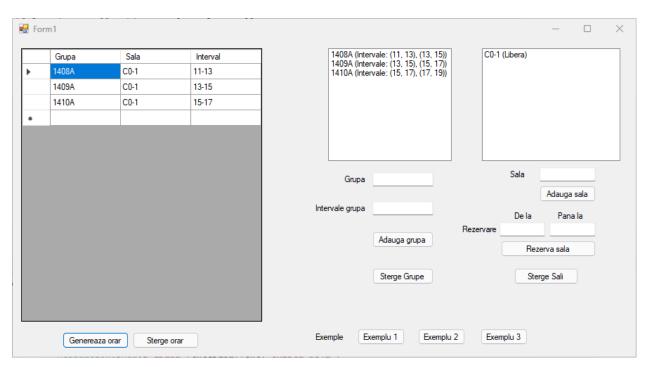
Adăugarea sălilor:



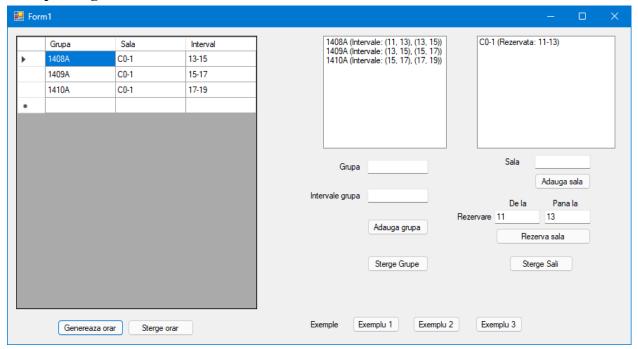
Rezervarea unei săli:



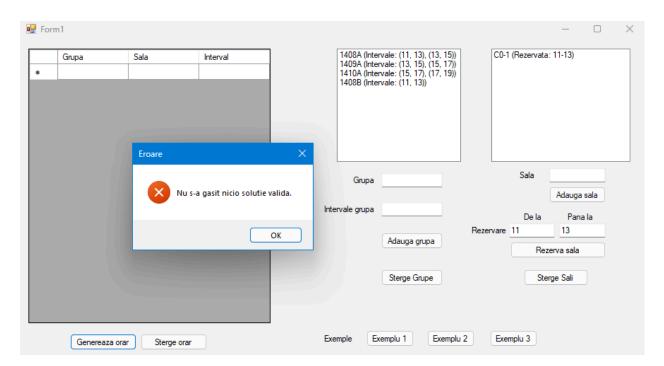
Exemplu de generare orar:



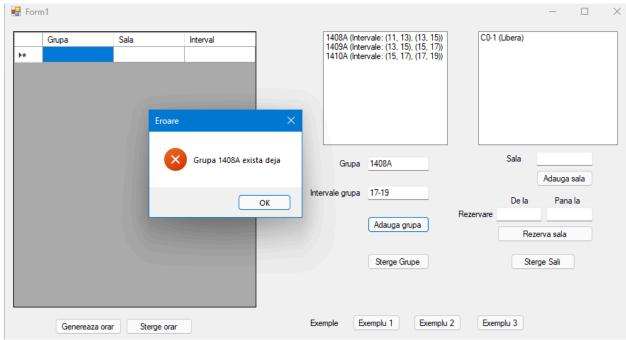
Exemplu de generare orar cu sală rezervată:



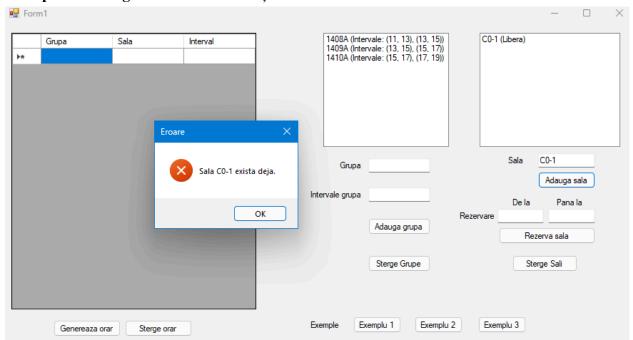
Exemplu de caz în care nu s-a găsit nicio soluție:



Exemplu de adăugare de grupă cu același nume:



Exemplu de adăugare de sală cu același nume:



6. Concluzii

Proiectul demonstrează utilizarea algoritmului Forward Checking pentru rezolvarea problemelor de alocare. Implementarea este eficientă și oferă flexibilitate utilizatorului prin adăugarea manuală de săli și grupe și prin posibilitatea de a genera și reseta orarul.

7. Bibliografie

- Florin Leon, "Inteligență artificială, Curs 3, Jocuri. Satisfacerea constrângerilor"
 Documentația oficială Microsoft C#