E.A.6.5 (Schur's Lemma)

1.1 Modellazione

1.1.1 Variabili

Dato $n \geq 1$ siano
- $\mathcal{N} = \{1, 2, 3, ..., n\}$ l'insieme dei numeri da 1 a n
- $\mathcal{U} = \{1, 2, 3\}$
- $X = \{X_i^u \mid i \in \mathcal{N} \wedge u \in \mathcal{U}\}$ l'insieme di variabili t.c.
  - X_i^u è vera se l'i-esimo numero sta nell'urna u

1.1.2 Vincoli

$$\phi = \phi_{\text{ogni\_numero\_in\_almeno\_un'urna}} \wedge \phi_{\text{ogni\_numero\_in\_al\_più\_un'urna}} \wedge \phi_{\text{aritmetica}}$$

$$\phi_{\text{ogni\_numero\_in\_almeno\_un'urna}} = \bigwedge_{i \in \mathcal{N}} \left(\bigvee_{u \in \mathcal{U}} X_i^u \right)$$

$$\phi_{\text{ogni\_numero\_in\_al\_più\_un'urna}} = \bigwedge_{\substack{i \in \mathcal{N} \\ u \in \mathcal{U} \\ t \in \mathcal{U} \\ u < t}} X_i^u \rightarrow \neg X_i^t$$

$$\phi_{\text{aritmetica}} = \bigwedge_{\substack{u \in \mathcal{U} \\ i \in \mathcal{N} \\ j \in \mathcal{N} \\ i < j \wedge \\ i+j \leq n}} \left(X_i^u \wedge X_j^u \right) \rightarrow \neg X_{i+j}^u$$

In particolare

$$\left(X_i^u \wedge X_j^u \right) \rightarrow \neg X_{i+j}^u =$$
$$\neg \left(X_i^u \wedge X_j^u \right) \vee \neg X_{i+j}^u =$$
$$\left(\neg X_i^u \vee \neg X_j^u \right) \vee \neg X_{i+j}^u =$$
$$\neg X_i^u \vee \neg X_j^u \vee \neg X_{i+j}^u =$$

1.2 Istanziazione

1.2.1 Variabili

Dato $n = 5$ si ha

– $X = \{X_1^1, X_1^2, X_1^3, X_2^1, X_2^2, X_2^3, X_3^1, X_3^2, X_3^3, X_4^1, X_4^2, X_4^3, X_5^1, X_5^2, X_5^3\}$

1.2.2 Vincoli

$$
\begin{aligned}
&\phi_{\text{ogni\_numero\_in\_almeno\_un'urna}} = (\\
&\quad (X_1^1 \vee X_1^2 \vee X_1^3) \wedge\\
&\quad (X_2^1 \vee X_2^2 \vee X_2^3) \wedge\\
&\quad (X_3^1 \vee X_3^2 \vee X_3^3) \wedge\\
&\quad (X_4^1 \vee X_4^2 \vee X_4^3) \wedge\\
&\quad (X_5^1 \vee X_5^2 \vee X_5^3)\\
&)
\end{aligned}
$$

$$
\begin{aligned}
&\phi_{\text{ogni\_numero\_in\_al\_più\_un'urna}} = (\\
&\quad (X_1^1 \to \neg X_1^2) \wedge (X_1^1 \to \neg X_1^3) \wedge (X_1^2 \to \neg X_1^3) \wedge\\
&\quad (X_2^1 \to \neg X_2^2) \wedge (X_2^1 \to \neg X_2^3) \wedge (X_2^2 \to \neg X_2^3) \wedge\\
&\quad (X_3^1 \to \neg X_3^2) \wedge (X_3^1 \to \neg X_3^3) \wedge (X_3^2 \to \neg X_3^3) \wedge\\
&\quad (X_4^1 \to \neg X_4^2) \wedge (X_4^1 \to \neg X_4^3) \wedge (X_4^2 \to \neg X_4^3) \wedge\\
&\quad (X_5^1 \to \neg X_5^2) \wedge (X_5^1 \to \neg X_5^3) \wedge (X_5^2 \to \neg X_5^3)\\
&)
\end{aligned}
$$

$$
\begin{aligned}
&\phi_{\text{aritmetica}} = (\\
&\quad (X_1^1 \wedge X_2^1 \to \neg X_3^1) \wedge (X_1^1 \wedge X_3^1 \to \neg X_4^1) \wedge (X_1^1 \wedge X_4^1 \to \neg X_5^1) \wedge\\
&\quad (X_2^1 \wedge X_3^1 \to \neg X_5^1) \wedge\\
&\quad (X_1^2 \wedge X_2^2 \to \neg X_3^2) \wedge (X_1^2 \wedge X_3^2 \to \neg X_4^2) \wedge (X_1^2 \wedge X_4^2 \to \neg X_5^2) \wedge\\
&\quad (X_2^2 \wedge X_3^2 \to \neg X_5^2) \wedge\\
&\quad (X_1^3 \wedge X_2^3 \to \neg X_3^3) \wedge (X_1^3 \wedge X_3^3 \to \neg X_4^3) \wedge (X_1^3 \wedge X_4^3 \to \neg X_5^3) \wedge\\
&\quad (X_2^3 \wedge X_3^3 \to \neg X_5^3) \wedge\\
&)
\end{aligned}
$$

1.3 Encoder

```
import it.uniroma1.di.tmancini.utils.*;
import it.uniroma1.di.tmancini.teaching.ai.SATCodec.*;
import java.util.*;

public class SchursToSAT {
  public static void main(String args[]) {
    int n = 32;

    var numbers = new IntRange("numbers", 1, n);
    var urns = new IntRange("urns", 1, 3);

    var encoder = new SATEncoder("Schurs", "schurs.cnf");
    encoder.defineFamilyOfVariables("X", numbers, urns);

    // Ogni numero in almeno un'urna
    for (var i : numbers.values() ) {
      for (var u : urns.values()) {
        encoder.addToClause("X", i, u);
      }
      encoder.endClause();
    }

    // Ogni numero in al più un'urna
    for (var i : numbers.values()) {
      for (var u : urns.values()) {
        for (int t = u + 1; t ≤ 3; t++) {
          encoder.addNegToClause("X", i, u);
          encoder.addNegToClause("X", i, t);
          encoder.endClause();
        }
      }
    }

    // Aritmetica
    for (var u : urns.values()) {
      for (int i = 1; i ≤ n; i ++) {
        for (int j = i + 1; i + j ≤ n; j ++) {
          encoder.addNegToClause("X", i, u);
          encoder.addNegToClause("X", j, u);
          encoder.addNegToClause("X", i + j, u);
          encoder.endClause();
        }
      }
    }

    encoder.end();
  }
}
```

1.4 Decoder

```
import it.uniroma1.di.tmancini.utils.*;
import it.uniroma1.di.tmancini.teaching.ai.SATCodec.*;
import java.util.*;

public class SATToSchurs {
  public static void main(String args[]) throws
java.io.IOException, java.io.FileNotFoundException {
    var decoder = new SATModelDecoder(args);
    decoder.run();

    int maxVar = decoder.getMaxVar();
    int n = maxVar / 3;

    ArrayList<ArrayList<Integer>> urns = new ArrayList();
    urns.add(new ArrayList<Integer>());
    urns.add(new ArrayList<Integer>());
    urns.add(new ArrayList<Integer>());

    for (int i = 1; i ≤ maxVar; i++) {
      Boolean v_i = decoder.getModelValue(i);
      if (v_i == null || !v_i) continue;

      SATModelDecoder.Var variable = decoder.decodeVariable(i);
      int number = variable.getIndices().get(0);
      int urn = variable.getIndices().get(1) - 1;

      urns.get(urn).add(number);
    }

    for (var urn : urns) {
      for (var number : urn) {
        System.out.print("" + number + ", ");
      }
      System.out.println();
    }
    System.out.println();
  }
}
```