UNIVERSITÉ DU LUXEMBOURG

Project 1: Design of Parallel Algorithms with Lattice Data Type

Lattice Theory for Parallel Programming

Goals

- ✯ Understanding a new model of computation based on lattice theory.
- ✯ Implementing two algorithms in this model on GPU.
- ✯ The code on your Github repository generated by clicking here: `https://classroom.github.com/a/lIhI8wMB`.
  Note: one team member first create the team, and then you join the team.
- ✯ **Deadline:** 16th of November 2024 (23h59).
- ✯ Don't share your code.
- ✯ This is a solo project or in a team of 3 students maximum.

1 Warm-up

Exercise 1 – Parallel Minimum

In C++, implement an algorithm to find the minimum in an array:

- Implement a sequential algorithm first.
- In CUDA/C++, implement a version using a global minimum variable (decreasing integer lattice) and the fixpoint model of computation.
- In CUDA/C++, implement a state-of-the-art reduction parallel algorithm on GPU (see *Chapter 10. Reduction And minimizing divergence* of the book *Programming Massively Parallel Processors A Hands-on Approach*.
- Use a massive number of threads (eg. 1024 blocks of 1024 threads) to compare the efficiency of both approaches.
- Benchmark on the HPC of the University using `hopper` nodes on IRIS.
- In the README.md, include benchmarking plot and an analysis of your results.

2 Parallel Simplification of Logical Formula

2.1 Constraint Network

In the following, we consider constraint programming over integer variables only. Let X be a finite set of variables and C be a finite set of constraints. Without loss of generality, we represent the domain of variables using intervals. Let $I = \{[\ell, u] \mid \ell \in \mathbb{Z}, u \in \mathbb{Z}, \ell \leq u\} \cup \{\bot\}$ be the set of intervals ordered by inclusion with a special element $\bot$ representing the empty interval. We define $lb([\ell, u]) \triangleq \ell$ and $ub([\ell, u]) \triangleq u$ to extract the lower and upper bounds. The intersection of intervals is defined as $[\ell, u] \cap [\ell', u'] \triangleq [\max\{\ell, \ell'\}, \min\{u, u'\}]$, and inclusion as $[\ell, u] \subseteq [\ell', u'] \Leftrightarrow \ell \geq \ell' \wedge u \leq u'$.

A *constraint network* is a pair $P = \langle d, C\rangle$ such that $d \in X \to I$ is the *domain function*. We denote $\mathbf{D}$ the set of all domain functions $X \to I$ ordered pointwise ($d \leq d' \Leftrightarrow \forall x \in X,\ d(x) \subseteq d'(x)$). An *assignment* is a map $asn : X \to \mathbb{Z}$, and we denote the set of all assignments by $\mathbf{Asn}$. The set of solutions of a constraint is given by $rel(c) \subseteq \mathbf{Asn}$. For instance, $rel(x + y \leq 5) \triangleq \{asn \in \mathbf{Asn} \mid asn(x) + asn(y) \leq 5\}$. The set of solutions of a constraint network is:

$$\begin{aligned} sol(d, C) \triangleq \{&asn \in \mathbf{Asn} \mid \\ &\forall c \in C,\ asn \in rel(c) \wedge \forall x \in X,\ asn(x) \in d(x)\}\end{aligned}$$

2.2 Ternary Constraint Network

A *ternary constraint network* (TCN) is a constraint network with only constraints of arity 3 such as $x = y + z$ and $b = (x \leq y)$ with $\{x, y, z, b\} \subseteq X$.

Definition 1. A ternary constraint network $\langle d, C\rangle$ is a constraint network such that each $c \in C$ is of the form $x = y \odot z$ where $x, y, z \in X$ are variables and $\odot \in \{+, *, min, max, =, \leq\}$ an arithmetic operator.

We write $\mathbf{TCN}$ the set of all ternary constraint networks. Unary constraints of the form $x = k$, $x \leq k$ and $x \geq k$ where $x \in X$ and $k \in \mathbb{Z}$ are directly represented as domains of the variables. The lack of subtraction is justified by the relational semantics of constraints as we can rewrite $x = y - z$ into $y = x + z$ without loss of precision. The semantics of the constraints is given as follows:

- $rel(x = y + z) \triangleq \{asn \in \mathbf{Asn} \mid asn(x) = asn(y) + asn(z)\}$
- $rel(x = y * z) \triangleq \{asn \in \mathbf{Asn} \mid asn(x) = asn(y) * asn(z)\}$
- $rel(x = \min(y, z)) \triangleq \{asn \in \mathbf{Asn} \mid asn(x) = \min(asn(y), asn(z))\}$
- $rel(x = \max(y, z)) \triangleq \{asn \in \mathbf{Asn} \mid asn(x) = \max(asn(y), asn(z))\}$
- $rel(x = y = z) \triangleq \{asn \in \mathbf{Asn} \mid asn(x) = (\!|\ asn(y) = asn(z)\ ?\ 1 \mathbin{⨾} 0\ |\!)\}$
- $rel(x = y \leq z) \triangleq \{asn \in \mathbf{Asn} \mid asn(x) = (\!|\ asn(y) \leq asn(z)\ ?\ 1 \mathbin{⨾} 0\ |\!)\}$

2.3 Equivalent Classes of Variables

Suppose the following constraint network:

$$\langle\{x \mapsto [0, 10], y \mapsto [0, 10], z \mapsto [5, 10], ONE \mapsto [1, 1]\}, \{z = x + y, ONE = z = x\}\rangle$$

We have an equality constraint $z = x$, therefore the constraint network could be simplified to:

$$\langle\{x \mapsto [5, 10], y \mapsto [0, 10], ONE \mapsto [1, 1]\}, \{x = x + y\}\rangle$$

We have removed the variable z, and substituted its occurrence in the constraint. Of course, to preserve equivalence between the initial constraint network and the simplified one, we must keep track of equivalent variables. We define a structure E to achieve this job[1].

We aim at finding a partition E of X such that each component Y of E represents a set of equivalent variables. More precisely, for each component $Y \in E$, all pairs of variables $x, y \in Y$ are connected by an equality constraint $x = y$.

[1] It might be useful to review *union-find algorithm*.

We write $[x]_E \in E$ the equivalence class of x in E. Note that E is a *set of sets*, hence $[x]_E$ is the set of all variables equivalent to x.

Example. Let us suppose $X = \{x, y\}$. Initially, we have not detected any equality, hence $E = \{\{x\}, \{y\}\}$, that is, x and y are in different equivalent classes. In particular, we have $[x]_E = \{x\}$ and $[y]_E = \{y\}$. If we detect $x = y$, then we can merge their equivalent classes and we obtain $E = \{\{x, y\}\}$. In that case, we have $[x]_E = \{x, y\}$ and $[y]_E = \{x, y\}$.

The equivalence classes are discovered by various preprocessing techniques. Initially, we suppose the variables are all distinct which is given by the partition $\mathit{init}(X) \triangleq \{\{x\} \mid x \in X\}$. We add a variable equality $x = y$ by removing both equivalence classes $[x]_E$ and $[y]_E$ from E and adding back their union into the partition.

$$\begin{aligned}
&\mathit{merge}(E, x, y) \triangleq \\
&\quad \textbf{let } XY = \{[x]_E\} \cup \{[y]_E\} \textbf{ in} \\
&\quad (E \setminus XY) \cup \{\textstyle\bigcup_{S \in XY} S\}
\end{aligned}$$

The interval domain of a variable $x \in X$ in an equivalence class Y is the intersection of all variables' domains in Y, defined as $d_E(x) \triangleq \bigcap_{y \in [x]_E} d(y)$.

We suppose variables are totally ordered (e.g. by an indexing) and we choose $\min Y$ to be the representative variable of the equivalence class Y.

2.4 Exercise

Exercise 2 – Parallel Simplification of Logical Formula

Your task is to provide a function $\mathit{simplify} : \mathbf{TCN} \to \mathbf{TCN} \times \mathcal{P}(\mathcal{P}(X))$ such that, for all $\langle d, C\rangle \in \mathbf{TCN}$ and $\mathit{simplify}(d, C) = (\langle d', C'\rangle, E)$:

- $E \subseteq \mathcal{P}(\mathcal{P}(X))$ is a partition of the variables as described above.
- The simplified TCN is equivalent:

$$\mathit{sol}(d, C) = \{\mathit{asn} \in \mathbf{Asn} \mid \mathit{asn} \in \mathit{sol}(d', C'), \forall x \in X,\ \mathit{asn}(x) \in d'_E(x)\}$$

A requirement is to design your algorithm as a *parallel fixpoint computation* over a lattice of your choice. At least, you should formalize and implement the detection of equality, that is, all constraint $x = y = z$ such that $d(x) = [1, 1]$.

There are many directions for improvements. For instance, notice that new equalities can be discovered using standard algebraic equivalence, e.g., $x = y + z$ with $d(z) = [0, 0]$ is the same as $x = y$.

Deliverables.

- If you do not present this project in-class, produce a short video (maximum 10 minutes), and give the link of the upload in the README of the project. Present the main ideas and the formalization.
- The formalization of your function *simplify* with proofs of correctness, in PDF file formatted using Latex.
- The parallel implementation of the algorithm on GPU, with benchmarking analysis and any other information you find relevant.
- You should benchmark your code on the following TCN instances[2]:
  `https://uniluxembourg-my.sharepoint.com/:u:/g/personal/pierre_talbot_uni_lu/EfgfQxdG0_BEjpm96LM1ihoBK6Yd6Om_ i4VPaEUgjTiPNg?e=0ozpR6`
  Of course, you can design smaller instances to test the correctness of your algorithm.
- Your program should be usable as follows:

  ```
  ./simplify -o output.tcn network.tcn
  ```

  You can add different options to enable/disable some optimizations, or propose different variants of the algorithm.

TCN Input Format. The constraint network $\langle\{x_0 \mapsto [1,3], x_1 \mapsto [1,1], x_2 \mapsto [0,3], x_3 \mapsto [0,2]\}, \{x_0 = x_1 + x_2, x_1 = x_2 \leq x_3, x_1 = x_0 = x_3\}$ is represented by the following input format:

```
4
1 3
1 1
0 3
0 2
3
0 1 + 2
1 2 L 3
1 0 = 3
```

The first line indicates the number of variables N. The N next lines are the domains of the variables (here we have $d(x_0) = [0,1]$). The next line is the number of constraints M. The M next lines are the constraints, for instance `0 1 + 2` represents the constraint $x_0 = x_1 + x_2$. The operators are represented by a single character in the set $\{+, *, A, I, L, =\}$ where `A` is for $\max$, `I` is for $\min$ and `L` is for $\leq$.

TCN Output Format. In the previous example, we can at least remove the equality constraint $x_1 = x_0 = x_3$ and the variable x_3.

```
4
1 2
1 1
0 3
0
3
0 1 + 2
1 2 L 3
R 1 0 = 3
```

In this example, we have the equivalence class $\{x_0, x_3\}$ and we choose the representative variable to be x_0. Instead of writting the domain of x_3, we write the index of its representative variable (here x_0). Note that the domain of x_0 is now $[1,2]$. Put the symbol `R` in front of the constraints that can be removed. Note that to simplify the assessment, you cannot add new constraints.

[2]They are generated from the MiniZinc challenge 2024 (`https://www.minizinc.org/challenge/2024/results/`).