Фізико-технічний інститут **Осінь 2021**

кафедра математичних методів захисту інформації

Теорія складності

2. Теорія обчислюваності

2.7 Зведення мов та задач

Досить часто при розв'язку певної задачі використовують відомий розв'язок іншої, але схожої задачі. Для того, щоб скористатися розв'язком іншої задачі, необхідно мати певну відповідність між задачами. *Зведенням* і називають певну процедуру (або алгоритм) перетворення однієї (обчислювальної) задачі в іншу.

Неформально, зведення задач використовується з метою використання відомих методів розв'язку. Тобто, якщо задача Π_1 зводиться до задачі Π_2, то з існування розв'язку для задачі Π_2 випливає існування розв'язку задачі Π_1. Таким чином зведення задач визначає певне бінарне відношення на множині всіх задач, при чому це відношення також дозволяє порівнювати складність задач. Якщо задача Π_1 зводиться до задачі Π_2 і задача Π_2 має певний розв'язок, то його можна використати для розв'язання задачі Π_1, але не завжди це буде правильним в зворотний бік. Це означає, що в такому випадку, можна сказати, що задача Π_2 має не меншу складність, ніж задача Π_1 (тобто, складність задачі Π_2 є такою ж як у задачі Π_1 або складність задачі Π_2 є більшою за складність задачі Π_1).

Зауваження. В математиці зведення задач є настільки розповсюдженим, що існує жарт (і не один) про різницю в методах розв'язку задач математиків та, наприклад, фізиків. Для розв'язку задачі "закип'ятити чайник" і фізик, і математик будуть використовувати однакову послідовність дій: налити воду у чайник, поставити чайник на плиту, включити плиту, довести воду кипіння. Але, якщо далі поставити задачу "закип'ятити чайник, повністю наповнений водою", то фізик зазвичай використає скорочену послідовність дій: поставити чайник на плиту, включити плиту, довести воду кипіння. Математик же зазвичай виллє воду з чайника і використає попередню послідовність дій, звівши задачу до попередньої. Зауважимо, що обидва методи розв'язують задачу "закип'ятити чайник,

повністю наповнений водою". Більш того, в певному розумінні обидва методи є оптимальними: фізик використовує мінімальну кількість дій для розв'язку, а математик максимально використовує вже наявні розв'язки, при цьому зменшуючи кількість міркувань та обґрунтувань коректності нової послідовності дій.

Також дуже часто вважають, що при зведенні більш складну задачу зводять до простішої задачі. Але в таких випадках в слово "зведення" вкладається дещо інший зміст. Так, якщо в попередньому жарті поставити дещо "складнішу" задачу "закип'ятити брудний чайник", то замість послідовності дій: помити чайник, налити воду у чайник, поставити чайник на плиту, включити плиту, довести воду кипіння, можна було б сказати, що достатньо помити чайник і звести задачу до попередньої. Але тут хоч під зведенням розуміють використання розв'язку іншої задачі, але фактично виконується декомпозиція задачі "закип'ятити брудний чайник" на складові й виконується зведення підзадачі, а не всієї задачі, якщо не вважати дію помити чайник частиною зведення.

Інший приклад — це задачі перевірити, чи ділиться задане натуральне число в десятковому записі на 5, та задача перевірити, чи дорівнює остання цифра натурального числа в десятковому записі 0 або 5. Може здатися, що перша задача виглядає складнішою, але її можна звести до другої задачі. Насправді, другу задачу, в свою чергу, можна так само звести до першої, тобто вони є однакової складності. Іноді зведення, яке використовується, є настільки потужним за обчисленням, що дозволяє звести більш складну задачу до більш простої, але завжди за таким самим зведенням більш проста задача буде зводитися до більш складної, що фактично означає, що ці задачі мають однакову складність відносно такого зведення.

Таким чином, завжди в загальному розумінні будемо вважати, що, якщо задача Π_1 зводиться до задачі Π_2, то задача Π_2 має не меншу складність (відносно такого зведення), оскільки використання зведення та розв'язку задачі Π_2 при розв'язанні задачі Π_1 є одним з можливих розв'язків, але можуть бути й інші, які будуть використовувати менше ресурсів. Так, якщо задача Π_2 є нерозв'язною і задача Π_1 зводиться до задачі Π_2, то необов'язково задача Π_1 буде також нерозв'язною, оскільки можуть існувати інші методи розв'язання цієї задачі, але задача Π_2 залишиться нерозв'язною.

Дійсно, розглянемо ще один приклад: задача Π_1 — задача знайти розв'язок квадратного рівняння $ax^2 + bx + c = 0$ для довільних дійсних чисел a, b і c, задача Π_2 — задача знайти розв'язок квадратного рівняння $x^2 + bx + 1 = 0$ для довільного дійсного числа b і задача Π_3 — задача знайти розв'язок квадратного рівняння $x^2 - 2x + 1 = 0$. Будь-хто, хто знає, як розв'язати задачу Π_1, зможе також розв'язати задачі Π_2 і Π_3. Алгоритмічний розв'язок задачі Π_2 навряд чи допоможе розв'язати задачу Π_1, але його можна використати для розв'язку задачі Π_3. І, нарешті, розв'язком задачі Π_3 може бути метод, який підставляє значення -1, 0 та 1 в рівняння і перевіряє чи є це значення коренем. Він точно не допоможе при розв'язанні задачі Π_1 та задачі Π_2 для довільних дійсних чисел a, b і c. З цього випливає, що задача Π_3 зводиться до задачі Π_2, яка в свою чергу зводиться до задачі Π_1, але не навпаки в загальному випадку.

Фактично зведення задач в неявному вигляді використовувалося при доведенні нерозв'язності задачі $HALT_\varepsilon$. Припускаючи, що існує розв'язок задачі $HALT_\varepsilon$ (вирішувач відповідної мови $HALT_\varepsilon$), будувався розв'язок задачі $HALT$ (вирішувач відповідної мови $HALT_{TM}$). Це означає, що використовувалося зведення задачі $HALT$ до задачі $HALT_\varepsilon$. Так само зведення використовувалося при доведенні невирішуваності мов E_{TM} та EQ_{TM}.

Приклад 2.1. В якості іншого прикладу розглянемо задачі

- $PRIME$ — для заданого натурального числа визначити, чи є воно простим;
- $COMPOSITE$ — для заданого натурального числа визначити, чи має воно нетривіальні дільники (відмінні від числа 1 та самого заданого числа);
- $FACTOR$ — для заданих натуральних чисел m і k визначити, чи має число m нетривіальний дільник, який є не більшим за число k.

Нехай існує (алгоритмічний) розв'язок задачі $PRIME$. Множина всіх натуральних чисел розбивається на множину простих чисел, множину складених чисел (які мають нетривіальний дільник) та множину $\{0, 1\}$. Для будь-якого натурального числа m можемо перевірити, чи дорівнює воно 0 або 1. Якщо воно дорівнює 0 або 1, то це число не є складеним, а, якщо не дорівнює одному з цих чисел, то використати для цього числа розв'язок задачі $PRIME$. Якщо розв'язок задачі $PRIME$ поверне ствердну відповідь для натурального числа m, то це означає, що число m є простим, а, отже, не є складеним. Якщо розв'язок задачі $PRIME$ поверне негативну відповідь для числа m, то це означає, що

число m не є простим, а, отже, є складеним. Описана послідовність дій є застосовною до довільного натурального числа і завжди повертає правильну відповідь. Таким чином, використовуючи розв'язок задачі $PRIME$, можна побудувати розв'язок задачі $COMPOSITE$. Це також означає, що побудовано зведення задачі $COMPOSITE$ до задачі $PRIME$. Побудова зведення не вимагає наявності розв'язку у задачі, до якої виконується зведення, оскільки зведення також виконує також функції порівняння складності задач. Так, якби не існувало розв'язку задачі $PRIME$, то зведення задачі $COMPOSITE$ до задачі $PRIME$ було б таким самим без використання розв'язку задачі $PRIME$. Але, навіть в такому випадку, поява будь-якого розв'язку задачі $PRIME$ означала б також розв'язок задачі $COMPOSITE$.

Аналогічно будується зведення задачі $PRIME$ до задачі $COMPOSITE$.

Обидві задачі $PRIME$ та $COMPOSITE$ зводяться до задачі $FACTOR$. Для цього треба взяти значення k рівним m в позначеннях задачі $FACTOR$ і повернути ту саму відповідь у випадку задачі $COMPOSITE$ та іншу відповідь у випадку задачі $PRIME$. Але жоден розв'язок задачі $PRIME$ чи $COMPOSITE$ не може розв'язати правильно задачу $FACTOR$ для довільних вхідних значень. Тобто, задача $FACTOR$ не зводиться ні до задачі $PRIME$, ні до задачі $COMPOSITE$.

Зведення задач як бінарне відношення є рефлексивним та транзитивним (тобто, є передпорядком або квазіпорядком на множині задач або відповідних мов). Дійсно, для довільного зведення розв'язок довільної задачі Π_1 якраз і розв'язує цю задачу, тобто завжди задача Π_1 зводиться сама до себе. І, якщо для довільного зведення та довільних задач Π_1, Π_2 та Π_3 мають місце зведення задачі Π_1 до задачі Π_2 та задачі Π_2 до задачі Π_3, то це означає, що розв'язок задачі Π_3 можна використати для розв'язання задачі Π_2, а це, в свою чергу, призведе до розв'язку задачі Π_1, а з цього випливає існування зведення задачі Π_1 до задачі Π_3. Саме тому властивості рефлексивності та транзитивності вважаються обов'язковими для будь-якого зведення, хоча теоретично можна придумати зведення, яке б порушувало ці властивості. В межах цього курсу будемо розглядати виключно зведення, які визначаються рефлексивним та транзитивним відношенням. Тому найбільш загальним означенням зведення буде таке означення.

Означення 2.1. *Зведенням* мов називають довільне бінарне відношення на множині всіх мов над алфавітом $\{0,1\}$, яке є рефлексивне та транзитивним. Мова $L_1 \subseteq \{0,1\}$ *зводиться* до мови $L_2 \subseteq \{0,1\}$, якщо впорядкована пара (L_1, L_2) належить бінарному відношенню, яке задає зведення. Для позначення зведення мов використовують символ $\leq$ з можливим використанням нижніх та верхніх індексів для уточнення конкретного зведення.

Наслідок. Обчислювальна задача Π_1 *зводиться* до обчислювальної задачі Π_2, якщо існують такі схеми кодування e_1 та e_2 задач Π_1 та Π_2 відповідно, що мова $L[\Pi_1, e_2]$ зводиться до мови $L[\Pi_2, e_2]$.

Зауваження. Іноді для позначення зведення використовують символ $\propto$, але таке позначення є дещо застарілим і далі будемо використовувати тільки символ $\leq$.

Зауваження. нову ж таки, не втрачаючи загальності, для формалізації поняття зведення будемо розглядати задачі розпізнавання і досліджувати властивості відповідних мов над алфавітом $\{0,1\}$, вважаючи, що мова $L_1 \subseteq \{0,1\}^*$ зводиться до мови $L_2 \subseteq \{0,1\}^*$ тоді й тільки тоді, коли зводяться масові задачі розпізнавання, що відповідають мовам L_1 та L_2 відповідно за певних схем кодування. Хоча формалізація поняття зведення дозволяє застосовувати його до довільних множин, включаючи формальні мови, без посилання на відповідні масові задачі.

З означення 2.1 випливає, що зведення не є єдиним, а існує багато різних видів зведення, які отримують, накладаючи певні додаткові обмеження на процедуру зведення мов (чи задач).

Означення 2.2. Нехай r — довільне зведення мов над алфавітом $\{0,1\}$, а L_1 та L_2 — довільні мови над алфавітом $\{0,1\}$. За допомогою запису $L_1 \leq_r L_2$ позначають твердження, що мова L_1 *зводиться за допомогою зведення* r до мови L_2, якщо це твердження є неправильним, то пишуть $L_1 \not\leq_r L_2$. За допомогою запису $L_1 <_r L_2$ позначають одночасне виконання умов $L_1 \leq_r L_2$ та $L_2 \not\leq_r L_1$. За допомогою запису $L_1|_r L_2$ позначають одночасне виконання умов $L_1 \not\leq_r L_2$ та $L_2 \not\leq_r L_1$. Мови L_1 і L_2 називають *еквівалентними відносно зведення* r (або r-*еквівалентними*), якщо $L_1 \leq_r L_2$ і $L_2 \leq_r L_1$, і позначають це як $L_1 =_r L_2$.

Бінарне відношення $=_r$ є еквівалентністю і окремий клас еквівалентності за відношенням $=_r$ називають *r-степенем*.

Наслідок. В свою чергу r-степені будь-якого зведення r є частково впорядкованими зведенням r.

Тобто, якщо $\mathbb{K}_1$ і $\mathbb{K}_2$ є r-степенями, то $\mathbb{K}_1 \leq \mathbb{K}_2$ тоді й тільки тоді, коли існують такі мови $L_1 \in \mathbb{K}_1$ і $L_2 \in \mathbb{K}_2$, що $L_1 \leq_r L_2$. Оскільки r-степінь є відношенням еквівалентності, то $\mathbb{K}_1 \leq \mathbb{K}_2$ тоді й тільки тоді, коли для будь-яких мов $L_1 \in \mathbb{K}_1$ і $L_2 \in \mathbb{K}_2$ є правильним твердження, що $L_1 \leq_r L_2$.

Оскільки існують різні зведення (або їх види), то це означає, що зведення також можна порівнювати між собою.

Означення 2.3. Нехай r_1 та r_2 є довільними зведеннями мов над алфавітом $\{ 0, 1 \}$. Зведення r_1 є *сильнішим ніж зведення* r_2 (відповідно зведення r_2 є *слабкішим ніж зведення* r_1), якщо для довільних мов $L_1, L_2 \subseteq \{ 0, 1 \}^*$ з умови $L_1 \leq_{r_1} L_2$ випливає, що $L_1 \leq_{r_2} L_2$. Зведення r_1 є *строго сильнішим ніж зведення* r_2 (відповідно зведення r_2 є *строго слабкішим ніж зведення* r_1), якщо зведення r_1 є сильнішим ніж зведення r_2 та існують такі мови $L_1, L_2 \subseteq \{ 0, 1 \}^*$, що $L_1 \leq_{r_2} L_2$, але $L_1 \not\leq_{r_1} L_2$. Зведення r_1 є *еквівалентним зведенню* r_2, якщо для довільних мов $L_1, L_2 \subseteq \{ 0, 1 \}^*$ твердження $L_1 \leq_{r_1} L_2$ є правильним тоді й тільки тоді, коли $L_1 \leq_{r_2} L_2$.

Наслідок. Оскільки кожне з довільних зведень r_1 та r_2 визначає транзитивне та рефлексивне бінарне відношення на множині мов і зведення r_1 є сильнішим ніж зведення r_2, то кожен r_2-степінь складається з одного або декількох (можливо, навіть, нескінченної кількості) r_1-степенів.

Означення 2.4. Нехай C — деякий клас складності (деяка множина) мов над алфавітом $\{ 0, 1 \}$, тобто підмножина булеану (множини всіх підмножин) $\mathcal{P}(\{ 0, 1 \}^*)$ (який ще позначається як $2^{\{ 0,1 \}^*}$), а r — деяке зведення мов над алфавітом $\{ 0, 1 \}$. Тоді клас складності C є *замкненим відносно зведення* r, якщо для довільної мови $L_c \subseteq \{ 0, 1 \}^*$ з класу складності C і довільної мови $L_1 \subseteq \{ 0, 1 \}^*$ з умови $L_1 \leq_r L_c$ випливає, що $L_1 \in C$. Мову L_1 називають *складною для класу складності* C (*C-складною*) *відносно зведення* r, якщо для кожної мови L_c з класу складності C виконується твердження $L_c \leq_r L_1$.

Мову L_1 називають *повною для класу складності* C (C*-повною*) *відносно зведення* r, якщо вона є складною для класу складності C відносно зведення r і належить класу складності C. Через $\leq_r (C)$ позначають множину всіх мов, які зводяться до мов класу складності C за допомогою зведення r, а для довільної мови $L_1 \subseteq \{0,1\}^*$ через $\leq_r (L_1)$ позначають множину всіх мов, які зводяться до мови L_1 за допомогою зведення r.

Наслідок. Клас складності ALL є замкненим відносно довільного зведення мов.

Перейдемо до конкретної побудови певних зведень та дослідимо їх властивості. Повертаючись до зведення масових обчислювальних задач, можна визначити один з найпростіших видів зведення як процедуру перетворення довільної індивідуальної задачі однієї масової задачі в певну індивідуальну задачу іншої масової задачі. При цьому, якщо обидві масові задачі є задачами розпізнавання, то таке перетворення має зберігати правильну відповідь для індивідуальних задач.

Якщо є зведення масової задачі розпізнавання Π_1 (з множиною індивідуальних задач D_{Π_1} та множиною індивідуальних задач з правильною відповіддю "так" $Y_{\Pi_1} \subseteq D_{\Pi_1}$) до масової задачі розпізнавання Π_2 (з множиною індивідуальних задач D_{Π_2} та множиною індивідуальних задач з правильною відповіддю "так" $Y_{\Pi_2} \subseteq D_{\Pi_2}$), то процедура зведення має бути таким відображенням множини D_{Π_1} в множину D_{Π_2}, що образ кожної індивідуальної задачі з множини Y_{Π_1} належить множині Y_{Π_2}, а образ кожної індивідуальної задачі з множини $D_{\Pi_1} \setminus Y_{\Pi_1}$ належить множині $D_{\Pi_2} \setminus Y_{\Pi_2}$.

Використовуючи цю ідею для зведення мов над алфавітом $\{0,1\}$ маємо таке формальне означення.

Означення 2.5. Нехай L_1 і L_2 — деякі мови над алфавітом $\{0,1\}$ і F — множина всюди визначених функцій виду $\{0,1\}^* \to \{0,1\}^*$, яка є замкненою відносно операції композиції функцій та містить тотожну функцію. Мова L_1 *зводиться за допомогою зведення функціонального типу з множиною функцій* F до мови L_2, якщо існує функція f в множині F (яку називають *функцією зведення*) така, що для довільного слова $x \in \{0,1\}^*$ $x \in L_1$ тоді й тільки тоді, коли $f(x) \in L_2$. При цьому мову L_1 називають *мовою, яка зводиться*, а мову L_2 називають *мовою,*

до якої зводять. Позначають таке зведення відносно множини функцій F як $L_1 \leq_F L_2$.

Наслідок. Для довільних мов $L_1, L_2 \subseteq \{0,1\}^*$ з умов, що мова L_1 зводиться до мови L_2 за допомогою зведення функціонального типу з множиною функцій F, і, що функцією цього зведення є функція $f \in F$, випливає, що $f(L_1) \subseteq L_2$ і $f(\overline{L_1}) \subseteq \overline{L_2}$.

Зауваження. Функція зведення необов'язково має бути ін'єктивною і може приймати всього лише два різних значення: одне для слів, що належать мові, яка зводиться, а інше для слів, що не належать мові, яка зводиться.

Наслідок. Нехай F є множиною функцій деякого зведення функціонального типу. Для довільних мов $L_1, L_2, L_3 \subseteq \{0,1\}$ виконується твердження, що, якщо функція $f \in F$ є функцією зведення мови L_1 до мови L_2, а функція $g \in F$ є функцією зведення мови L_2 до мови L_3, то функція $f \circ g$ є функцією зведення мови L_1 до мови L_3.

$\triangleright$ Якщо функція $f \in F$ є функцією зведення мови L_1 до мови L_2, то за означенням 2.5 з цього випливає, що для довільного слова $x \in \{0,1\}^*$ $x \in L_1$ тоді й тільки тоді, коли $f(x) \in L_2$. Аналогічно з того, що функція $g \in F$ є функцією зведення мови L_2 до мови L_3, випливає, що для довільного значення $x \in \{0,1\}^*$ $x \in L_2$ тоді й тільки тоді, коли $g(x) \in L_3$. Оскільки множина функцій F є замкненою відносно операції композиції функцій, то функція $f \circ g$ також належить множині функцій F. Для довільного слова $x \in \{0,1\}^*$ це слово буде належати мові L_1 тоді й тільки тоді, коли $f(x) \in L_2$. В свою чергу слово $f(x)$ буде належати мові L_2 тоді й тільки тоді, коли $g(f(x)) \in L_3$. З цього випливає, що для довільного слова $x \in \{0,1\}^*$ це слово буде належати мові L_1 тоді й тільки тоді, коли $g(f(x)) \in L_3$. Це означає, що функція $f \circ g$ є функцією зведення мови L_1 до мови L_3. $\square$

Наслідок. Всі зведення функціонального типу є рефлексивними і транзитивними відношеннями на множині мов над алфавітом $\{0,1\}$.

Зауваження. Вимога до множини функцій F бути замкненою відносно операції композиції функцій є обов'язковою для гарантування властивості транзитивності зведення функціонального типу. А вимога, що множина функцій F має

містити тотожну функцію, гарантує, що будь-яке зведення функціонального типу є рефлексивним.

Теорема 2.1. *Для довільних мов $L_1, L_2 \subseteq \{0,1\}^*$ і довільної множини F всюди визначених функцій виду $\{0,1\}^* \to \{0,1\}^*$, яка є замкненою відносно операції композиції функцій та містить тотожну функцію, з умови $L_1 \leq_F L_2$ випливає, що $\overline{L_1} \leq_F \overline{L_2}$.*

$\triangleright$ Якщо $L_1 \leq_F L_2$, то з цього випливає, що існує всюди визначена функція $f \in F$ така, що для довільного слова $x \in \{0,1\}^*$ це слово належить мові L_1 тоді й тільки тоді, коли слово $f(x)$ належить мові L_2. Ця ж функція f є функцією зведення мови $\overline{L_1}$ до мови $\overline{L_2}$. Дійсно, для довільного слова $x \in \{0,1\}^*$, якщо $x \in L_1$, то $f(x) \in L_2$, але при цьому $x \notin \overline{L_1}$ і $f(x) \notin \overline{L_2}$. Якщо ж $x \notin L_1$, то $f(x) \notin L_2$, але при цьому $x \in \overline{L_1}$ і $f(x) \in \overline{L_2}$. Тобто, для довільного слова $x \in \{0,1\}^*$ це слово належить мові $\overline{L_1}$ тоді й тільки тоді, коли слово $f(x)$ належить мові $\overline{L_2}$. Це означає, що має місце зведення $\overline{L_1} \leq_m \overline{L_2}$ і функція f є функцією зведення мови $\overline{L_1}$ до мови $\overline{L_2}$ за допомогою зведення функціонального типу з множиною функцій F. $\square$

Приклад 2.2. Множина всіх всюди визначених функцій виду $\{0,1\}^* \to \{0,1\}^*$, множина всіх обчислювальних всюди визначених функцій виду $\{0,1\}^* \to \{0,1\}^*$, множина всіх всюди визначених функцій виду $\{0,1\}^* \to \{0,1\}^*$, які обчислюються машинами Тюрінга за поліноміальний час, та множина всіх всюди визначених функцій виду $\{0,1\}^* \to \{0,1\}^*$, які обчислюються машинами Тюрінга за лінійний час, є прикладами множин функцій, що є замкненими відносно операції композиції функцій та містять тотожну функцію., тобто множинами функцій відповідних зведень функціонального типу.

Вправа 2.1. Доведіть, що множина всюди визначених обчислювальних функцій $\{0,1\}^* \to \{0,1\}^*$ є замкненою відносно операції композиції функцій.

Згадуючи, що поняття зведення вводилося, в першу чергу, з метою використання наявних розв'язків для розв'язання інших задач, для практичного застосування зазвичай використовують зведення функціонального типу лише відносно

певної підмножини обчислювальних функцій. Одним з найбільш сильних зведень функціонального типу з використанням лише обчислювальних функцій є таке.

Означення 2.6. *m-зведенням* (від англ. many-one і англ. mapping) називають зведення функціонального типу відносно множини всіх всюди визначених обчислювальних функцій. m-зведення мови $L_1 \subseteq \{0,1\}^*$ до мови $L_2 \subseteq \{0,1\}^*$ позначають як $L_1 \leq_m L_2$.

Властивості. (m-*зведення*)

1. Для довільних мов $L_1, L_2 \subseteq \{0,1\}^*$, якщо $L_1 \leq_m L_2$ і L_2 є вирішуваною мовою, то мова L_1 також є вирішуваною.

   ▷ Якщо L_2 є вирішуваною мовою, то існує така машина Тюрінга M_2, що має місце тотожність $M_2(x) = L_2(x)$ для всіх слів $x \in \{0,1\}^*$, тобто вирішувач мови L_2. Якщо $L_1 \leq_m L_2$, то має існувати така обчислювальна та всюди визначена функція $f\colon \{0,1\}^* \to \{0,1\}^*$, що для довільного слова $x \in \{0,1\}^*$ має виконуватися умова: $x \in L_1$ тоді й тільки тоді, коли $f(x) \in L_2$. Оскільки функція f є обчислювальною, то має існувати машина Тюрінга M_f така, що для довільного слова $x \in \{0,1\}^* \quad M_f(x) = f(x)$. З існування таких машин Тюрінга M_2 та M_f випливає існування машини Тюрінга M, яка є композицією M_2 та M_f, тобто для довільного слова $x \in \{0,1\}^* \quad M(x) = M_2(M_f(x))$. Отже, для довільного слова $x \in \{0,1\}^*$ $M(x) = M_2(M_f(x)) = M_2(f(x)) = L_2(f(x)) = L_1(x)$, а це означає, що машина Тюрінга M вирішує мову L_1, тобто, мова L_1 є вирішуваною. □

2. Для довільних мов $L_1, L_2 \subseteq \{0,1\}^*$, якщо $L_1 \leq_m L_2$ і L_1 є невирішуваною мовою, то мова L_2 також є невирішуваною.

   ▷ Скористаємося методом від супротивного. Припустимо, що мова L_2 є вирішуваною. Тоді, з умови $L_1 \leq_m L_2$ та попередньої властивості m-зведення випливає, що мова L_1 має бути вирішуваною, а за умовою мова L_1 є невирішуваною. Суперечність. Отже, якщо $L_1 \leq_m L_2$ і L_1 є невирішуваною мовою, то мова L_2 також є невирішуваною. □

3. Для довільних мов $L_1, L_2 \subseteq \{0,1\}^*$, якщо $L_1 \leq_m L_2$ і L_2 є рекурсивно зліченною мовою, то мова L_1 також є рекурсивно зліченною.

   ▷ Доведення є дуже схожим на доведення першої властивості.

   Якщо L_2 є рекурсивно зліченною мовою, то існує така машина Тюрінга M_2,

що має місце рівність $M_2(x) = 1$ для всіх слів $x \in L_2$, тобто розпізнавач мови L_2. Якщо $L_1 \leq_m L_2$, то має існувати така обчислювальна та всюди визначена функція $f\colon \{0,1\}^* \to \{0,1\}^*$, що для довільного слова $x \in \{0,1\}^*$ має виконуватися умова: $x \in L_1$ тоді й тільки тоді, коли $f(x) \in L_2$. Оскільки функція f є обчислювальною, то має існувати машина Тюрінга M_f така, що для довільного слова $x \in \{0,1\}^*$ $M_f(x) = f(x)$. З існування таких машин Тюрінга M_2 та M_f випливає існування машини Тюрінга M, яка є композицією M_2 та M_f, тобто для довільного слова $x \in \{0,1\}^*$ $M(x) = M_2(M_f(x))$. Отже, для довільного слова $x \in L_1$ машина Тюрінга M_f при застосуванні до вхідного слова x поверне слово $f(x)$, яке належить мові L_2, а отже машина Тюрінга M_2 обов'язково зупиниться і прийме таке вхідне слово $f(x)$, оскільки слово $f(x)$ належить мові L_2, а машина Тюрінга M_2 є розпізнавачем мови L_2. Таким чином машина Тюрінга M обов'язково приймає довільне слово з мови L_1, а це означає, що вона є розпізнавачем мови L_1, а мова L_1 є рекурсивно зліченною. □

4. Для довільних мов $L_1, L_2 \subseteq \{0,1\}^*$, якщо $L_1 \leq_m L_2$ і L_2 є корекурсивно зліченною мовою, то мова L_1 також є корекурсивно зліченною.

   ▷ Якщо $L_1 \leq_m L_2$, то за теоремою 2.1 з цього випливає, що $\overline{L_1} \leq_m \overline{L_2}$. За умовою мова L_2 є корекурсивно зліченною, отже мова $\overline{L_2}$ є рекурсивно зліченною. Тоді з попередньої властивості випливає, що і мова $\overline{L_1}$ також є рекурсивно зліченною. А це означає, що мова L_1 є корекурсивно зліченною. □

Наслідок. Класи складності R, RE та $coRE$ є замкненими відносно m-зведення.

Також існують зведення функціонального типу, в яких висуваються додаткові вимоги до функції зведення.

Означення 2.7. 1-*зведенням* (англ. 1-reduction, one-one reduction) мови $L_1 \subseteq \{0,1\}^*$ до мови $L_2 \subseteq \{0,1\}^*$ називають зведення функціонального типу відносно множини всіх всюди визначених обчислювальних функцій, якщо існує ін'єктивна всюди визначена обчислювальна функція $f\colon \{0,1\}^* \to \{0,1\}^*$ така, що для довільного слова $x \in \{0,1\}^*$ $x \in L_1$ тоді й тільки тоді, коли $f(x) \in L_2$. 1-зведення мови $L_1 \subseteq \{0,1\}^*$ до мови $L_2 \subseteq \{0,1\}^*$ позначають як $L_1 \leq_1 L_2$.

Наслідок. m-зведення є слабкішим за 1-зведення.

▷ Якщо для довільних мов $L_1, L_2 \subseteq \{0,1\}^*$ мова L_1 зводиться за допомогою 1-зведення до мови L_2 і функція $f\colon \{0,1\}^* \to \{0,1\}^*$ є функцією цього зведення, то мова L_1 також зводиться за допомогою m-зведення до мови L_2 з функцією f, оскільки 1-зведення є частковим випадком m-зведення (з додатковою вимогою ін'єктивності функції зведення). Тобто, для довільних мов $L_1, L_2 \subseteq \{0,1\}^*$ з твердження $L_1 \leq_1 L_2$ завжди випливає, що $L_1 \leq_m L_2$. □

Фактично головна ідея зведення функціонального типу, зокрема m-зведення, у використанні відображення індивідуальних задач однієї масової задачі у множину індивідуальних задач іншої масової задачі із збереженням правильної відповіді. З іншого боку для розв'язку однієї індивідуальної задачі можуть знадобитися розв'язки декількох інших індивідуальних задач, а не тільки однієї. І не обов'язково, щоб ці індивідуальні задачі мали однакову правильну відповідь. Сама процедура зведення може опрацювати всі отримані відповіді для свого розв'язку.

Для формалізації такого підходу використовують поняття *оракулу*. Кажуть, що певний оракул розв'язує масову задачу (функціональну задачу чи задачу розпізнавання), якщо він завжди дає правильну відповідь на будь-яку її індивідуальну задачу. Ця задача може бути невирішуваною, а сам оракул не повинен бути машиною Тюрінга чи іншим обчислювальним пристроєм. Під оракулом розуміють певну сутність, яку зазвичай розглядають як деякий "чорний ящик", що вміє розв'язувати одну масову задачу. За деяких умов розглядають оракулів, що можуть помилятися, але далі будемо вважати, що оракул завжди виконує свої обчислення без помилок. Подання на вхід оракула певної індивідуальної задачі будемо називати *запитом* (до оракула).

З точки зору зведення задач використання оракулу може бути корисним. Так, можна розглянути зведення масової задачі Π_1 до масової задачі Π_2, яка представлена у вигляді оракулу. Саме зведення полягає в побудові для кожної індивідуальної задачі Π_1 набору запитів до оракулу, який розв'язує задачу Π_2, та обробці отриманих від оракулу результатів. Якщо кожна індивідуальна задача Π_2, що входить до набору запитів матиме розв'язок, як і сама побудова запитів, то й індивідуальна задача Π_1 матиме розв'язок. Якщо всі індивідуальні задача масової задачі Π_2 матимуть розв'язок і кожна побудова запитів, то матиме

розв'язок масова задача Π_1. Якщо кількість запитів та самі запити до оракулу будуються до безпосереднього використання оракулу, то такий набір запитів називають *неадаптивною системою запитів*. Якщо ж кожен наступний запит будується в залежності від отриманих відповідей оракула на попередні запити, то такий набір запитів називають *адаптивною системою запитів*.

Без втрати загальності обмежимося оракулами тільки для задач розпізнавання. Тоді, з кожним таким оракулом можна ототожнити певну мову і вважати, що задано оракул, якщо визначена мова, яку він вирішує.

Для формального опису використання оракулу з адаптивною системою запитів корисною є таке узагальнення машини Тюрінга.

Означення 2.8. *Машиною Тюрінга з оракулом* називають абстрактний обчислювальний пристрій, який визначається кортежем $(k, \Gamma, \Sigma, \#, Q, Q_F, q_0, q^O_{query}, q^O_{yes}, q^O_{no}, \delta)$, де

$k \in \mathbb{N}$, $k \geq 3$ — кількість стрічок машини Тюрінга;

Γ — алфавіт машини Тюрінга або алфавіт стрічки;

$\# \in \Gamma$ — порожній символ;

$\Sigma \subseteq \Gamma \setminus \{\#\}$ — вхідний алфавіт;

Q — множина внутрішніх станів;

$Q_F \subseteq Q$ — множина кінцевих внутрішніх станів;

$q_0 \in Q$ — початковий стан;

$q^O_{query} \in Q$ — виділений внутрішній стан запиту до оракулу;

$q^O_{yes} \in Q$, $q^O_{yes} \neq q^O_{query}$, — виділений внутрішній стан відповіді оракула 'так';

$q^O_{no} \in Q$, $q^O_{no} \neq q^O_{query}$, $q^O_{no} \neq q^O_{yes}$, — виділений внутрішній стан відповіді оракула 'ні';

$\delta\colon Q \setminus (Q_F \cup \{q^O_{query}\}) \times \Gamma^k \nrightarrow Q \times \Gamma^{k-1} \times \{ L, S, R \}^k$ — функція переходів.

В машині Тюрінга з оракулом є виділена робоча стрічка (наприклад, передостання), яку називають стрічкою оракула. Під час роботи машини Тюрінга записує символи на цю стрічку. Як тільки внутрішній стан машини Тюрінга стає виділеним станом запиту до оракулу q^O_{query} умовно починає свою роботу оракул, який переглядає стрічку оракула і зчитує слово, яке на ній написано. Якщо це слово належить мові оракула, то наступним внутрішнім станом машини Тюрінга буде q^O_{yes}. Якщо це слово не належить мові оракула, то наступним внутрішнім станом машини Тюрінга буде q^O_{no}. Якщо слово не є словом над алфавітом оракула,

то машина Тюрінга зациклюється. Після відповіді оракула його стрічка буде містити лише порожні комірки і зчитувальні пристрої не змінюють свої позиції на жодній стрічці. З точки зору машини Тюрінга перехід в один із станів q^O_{yes} або q^O_{no} зі стану q^O_{query} виконуватиметься рівно один такт, не зважаючи на те, скільки часу необхідно для роботи оракулу.

Якщо мова оракула є відомою (без втрати загальності будемо вважати, що вона визначена над алфавітом $\{0,1\}$), то машину Тюрінга M з оракулом, який визначається мовою $O \subseteq \{0,1\}^*$ позначають як M^O.

Означення 2.9. Мова $L_1 \subseteq \{0,1\}^*$ *зводиться за Тюрінгом* до мови $L_2 \subseteq \{0,1\}^*$ (*за час* $T(n)$), якщо існує машина Тюрінга з оракулом, яка вирішує мову L_1 (за час $T(n)$) з використанням мови L_2 як мови оракула, і позначають це як $L_1 \leq_T L_2$.

Твердження 2.1. Зведення за Тюрінгом є слабкішим за m-зведення.

$\triangleright$ Нехай для довільних мов $L_1, L_2 \subseteq \{0,1\}^*$ має місце зведення $L_1 \leq_m L_2$. З цього випливає, що існує обчислювальна всюди визначена функція $f\colon \{0,1\}^* \to \{0,1\}^*$, яка є функцією зведення мови $L_1 \leq_m L_2$ до мови L_2. що для довільного слова $x \in \{0,1\}^*$ має виконуватися умова: $x \in L_1$ тоді й тільки тоді, коли $f(x) \in L_2$ $L_1 \leq_T L_2$ Оскільки функція f є обчислювальною, то має існувати машина Тюрінга M_f, яка обчислює цю функцію, тобто така, що для довільного слова $x \in \{0,1\}^*$ $M_f(x) = f(x)$. Якщо модифікувати машину Тюрінга M_f, додавши нову вихідну стрічку, а при цьому вважати її вихідну стрічку стрічкою оракула, роботу організувати так, що для довільного вхідного слова виконуються обчислення машини Тюрінга M_f, а потім робиться запит до оракулу з мовою L_2 щодо належності отриманого слова до мови L_2. Якщо оракул повертає ствердну відповідь, то модифікована машина Тюрінга $\tilde{M}_f$ також буде повертати ствердну відповідь; якщо оракул повертає заперечну відповідь, то машина Тюрінга $\tilde{M}_f$ також буде повертати заперечну відповідь. Побудована таким чином машина Тюрінга $\tilde{M}_f$ є машиною Тюрінга з оракулом L_2, тобто $\tilde{M}_f^{L_2}$. Для довільного слова $x \in \{0,1\}^*$ слово $f(x)$ належить мові L_2 тоді й тільки тоді, коли слово x належить мові L_1. Тобто, слово x належить мові L_1 тоді й тільки тоді, коли оракул, а слід за ним і машина Тюрінга $\tilde{M}_f^{L_2}$, повернуть ствердну відповідь. Отже машина Тюрінга $\tilde{M}_f^{L_2}$ вирішує мову L_1, а це означає, що мова L_1 зводиться за Тюрінгом до мови L_2. Фактично m-зведення є частковим випадком зведення за Тюрінгом,

коли робиться тільки один запит до оракулу і повертається відповідь, яку дасть оракул. □

Якщо розглянути неадаптивну систему запитів, то використання машини Тюрінга з оракулом є недоцільним (треба вводити додаткові обмеження). При неадаптивній системі запитів до оракулу для кожного вхідного слова тільки за його значенням має бути визначеними кількість запитів та які саме запити будуть зроблені. Щоб описати зведення, яке використовує таку неадаптивну систему запитів, необхідним є таке означення.

Означення 2.10. Мова $L_1 \subseteq \{0,1\}^*$ зводиться до мови $L_2 \subseteq \{0,1\}^*$ за допомогою *табличного зведення або tt-зведення* (англ. *truth-table reduction* або *tt-reduction*), якщо існують обчислювані всюди визначені функції $f_1\colon \{0,1\}^* \to \mathbb{N}$, $f_2\colon \{0,1\}^* \to \bigcup\limits_{k\in\mathbb{N}} (\{0,1\}^*)^k$ та $f_3\colon \{0,1\}^* \to U$, де U — множина всіх булевих функцій, тобто всіх відображень $\{0,1\}^k \to \{0,1\}$ для всіх чисел $k \in \mathbb{N}$, такі, що для довільного слова $x \in \{0,1\}^*$ $f_3(x) = \varphi(b_1, \ldots, b_{f_1(x)})$, де φ є булевою функцією від $f_1(x)$ булевих змінних виду $\{0,1\}^{f_1(x)} \to \{0,1\}$, а $f_2(x) = (z_1, \ldots, z_{f_1(x)})$, де $z_1, \ldots, z_{f_1(x)} \in \{0,1\}^*$, і $x \in L_1$ тоді й тільки тоді, коли $\varphi(L_2(z_1), \ldots, L_2(z_{f_1(x)})) = 1$. Позначають це через запис $L_1 \leq_{tt} L_2$.

Фактично tt-зведення можна розглядати як неадаптивну систему запитів до оракулу, де самі запити до оракулу та обробка відповідей оракулу не залежить від самих відповідей. Дійсно, функція f_1 відповідає за кількість запитів для кожного слова, функція f_2 — за побудову конкретних запитів (слів, щодо яких оракул має відповісти, чи належать вони мові L_2), а функція f_3 — за обробку результатів цих запитів та обчислення результату. Якщо зробити послаблення щодо попередньої обробки результатів, але не побудови конкретних запитів, то можна отримати дещо слабкіше зведення.

Означення 2.11. Мова $L_1 \subseteq \{0,1\}^*$ зводиться до мови $L_2 \subseteq \{0,1\}^*$ за допомогою *слабкого табличного зведення або wtt-зведення* (англ. *weak truth-table reduction* або *wtt-reduction*), якщо існують машина Тюрінга M з оракулом L_2 та обчислювані всюди визначені функції $f_1\colon \{0,1\}^* \to \mathbb{N}$ та $f_2\colon \{0,1\}^* \to \bigcup\limits_{k\in\mathbb{N}} (\{0,1\}^*)^k$ такі, що для довільного вхідного слова $x \in \{0,1\}^*$ машина Тюрінга M з оракулом L_2 зробить точно $f_1(x)$ запитів $(z_1, \ldots, z_{f_1(x)})$ до оракулу, які визначаються значенням $f_2(x) = (z_1, \ldots, z_{f_1(x)})$, де $z_1, \ldots, z_{f_1(x)} \in$

$\{0,1\}^*$, і при цьому машина Тюрінга M з оракулом L_2 вирішує мову L_1. Позначають це через запис $L_1 \leq_{wtt} L_2$.

Твердження 2.2. Зведення за Тюрінгом є слабкішим за wtt-зведення. wtt-зведення є слабкішим за tt-зведення. tt-зведення є слабкішим за m-зведення.

Означення 2.12. Мова $L_1 \subseteq \{0,1\}^*$ *обмежено зводиться за Тюрінгом* до мови $L_2 \subseteq \{0,1\}^*$ (*за час* $T(n)$) (або *зводиться за допомогою обмеженого зведення за Тюрінгом*), якщо існують всюди визначена обчислювальна функція $f\colon \mathbb{N} \to \mathbb{N}$ і машина Тюрінга з оракулом, яка вирішує мову L_1 (за час $T(n)$) з використанням мови L_2 як мови оракула, причому довжина кожного запиту до оракула не перевищує значення $f(|x|)$ для будь-якого вхідного слова $x \in \{0,1\}^*$, і позначають це як $L_1 \leq_{bT} L_2$.

Твердження 2.3. Обмежене зведення за Тюрінгом є еквівалентним wtt-зведенню.

З урахуванням всіх зведень, які розглядалися, маємо таку низку тверджень для довільних мов $L_1, L_2 \subseteq \{0,1\}^*$: $L_1 \leq_1 L_2 \Rightarrow L_1 \leq_m L_2 \Rightarrow L_1 \leq_{tt} L_2 \Rightarrow L_1 \leq_{wtt} L_2 \Rightarrow L_1 \leq_T L_2$.

Означення 2.13. Класи еквівалентності за зведенням за Тюрінгом називають *степенем Тюрінга* (англ. Turing degree) або *степенем нерозв'язності* (англ. degree of unsolvability). Для довільної мови $L_1 \subseteq \{0,1\}^*$ через $[L_1]$ позначають степінь Тюрінга, якому належить мова L_1. Множину всіх степенів Тюрінга позначають як $\mathcal{D}$.

Зведення за Тюрінгом також породжує відношення часткового порядку на множині $\mathcal{D}$. Існує єдиний степінь Тюрінга, який містить всі вирішувані мови. Він є найменшим в множині $\mathcal{D}$ і його позначають як $\mathbf{0}$.

Степені Тюрінга використовують, щоб визначити, які мови є більш невирішуваними. Степені зведень, які є сильнішими за зведення за Тюрінгом, використовують, коли є потрібним розбиття множини всіх мов на менші класи еквівалентності.