სირთულის თეორიის მოკლე მიმოხილვა

ალექსანდრე გამყრელიძე

სარჩევი

„სირთულის თეორია” *ამოცანებისა და მათი გადაწყვეტისათვის საჭირო ალგორითმების* სირთულეს შეისწავლის.
იმისათვის, რომ თვითონ ალგორითმი შევაფასოთ, აუცილებელია ისეთი კრიტერიუმების შემუშავება, რომლის დროსაც შეფასდება თვით ალგორითმის ხარისხი - მისი სისწრაფისა და მეხსიერების ხარჯვის რიგი. თანაც ეს კრიტერიუმები არ უნდა იყოს დამოკიდებული ამა თუ იმ გამომთვლელზე, რომელზედაც ალგორითმს გავუშვებთ -- ელემენტარული ოპერაციების ჩატარების სისწრაფეზე, მეხსიერების ზომაზე და ა.შ. ამიტომ ალგორითმისა და ამოცანის სირთულეს ითვლიან გამომთვლელთა თეორიული მოდელის -- ტიურინგის მანქანის გამოყენებით, რომელსაც უსასრულოდ დიდი მეხსიერება აქვს, ანუ რესურსებში შეზღუდული არ არის და რომელსაც განსაზღვრული აქვს რამოდენიმე ელემენტრული ოპერაცია, რისი შესრულების დროც ერთის ტოლადაა მიჩნეული. ამით იხსნება საკმარისი მეხსიერების არქონის გამო შექმნილი პრობლემები. ამრიგად, ალგორითმის სირთულე შესაბამისი ტიურინგის მანქანის მიერ შესრულებული ელემენტარული ოპერაციების რიცხვის ტოლია და თვით ალგორითმი შესაბამისი ტიურინგის მანქანის ტოლფასია (როგორც აღმოჩნდა, ყოველ ალგორითმს ერთი ტიურინგის მანქანა შეესაბამება და პირიქით - ყოველ ტიურინგის მანქანას ერთი ალგორითმი შეესაბამება, ასე რომ ეს ორი რამ ერთმანეთის ექვივალენტურია).

1 სირთულის თეორიის ელემენტები

1.1 რთული, მარტივი, გამოთვლადი და არაგამოთვლადი ამოცანები

უხეშად, ამოცანები ორ კლასად შეიძლება დაიყოს: ისეთები, რომლისთვისაც ალგორითმული ამოხსნა არ არსებობს (ანუ ამ ამოცანების გადაჭრა სასრულ დროში არ შეიძლება - მათი ყველა ალგორითმი ჩაიციკლება) და ისეთებად, რომლებისთვისაც ალგორითმები არსებობს. ისეთ ამოცანებში, რომლებისთვისაც ალგორითმები არსებობს, გამოყოფენ ამოცანათა სირთულის რამოდენიმე კლასს. აღმოჩნდა, რომ ერთ-ერთ კლასში ისეთი ამოცანებია, რომელთათვისაც რეალურ დროში ამოხსნა არსებობს, ხოლო მეორეში - ისეთები, რომელთათვისაც რეალურ დროში ამოხსნა დღეისათვის ცნობილი არ არის (ეს მკაცრად შემდგომში იქნება განსაზღვრული). აქვე უნდა აღინიშნოს, რომ ეს ამოცანები ამ ორ კლასში ზემოთ აღნიშნული კრიტერიუმებით არ განლაგებულა - ეს უფრო თეორიული შედეგების მიერ გამოწვეული დამთხვევაა.

ერთ-ერთი ყველაზე ძველი ამოცანა, რომელსაც ალგორითმული ამოხსნა არ აქვს (გარკვეული რესურსებით შეზღუდვის გათვალისწინებით), ანტიკური საბერძნეთიდან მოდის და რამოდენიმე ათასი წელი გადაუჭრელი რჩებოდა - ფიგურების ფარგლითა და სახაზავით აგების ალგორითმების პოვნა. როგორც XIX საუკუნეში მათემატიკოსებმა დაამტკიცეს, შეუძლებელია ისეთი ალგორითმების პოვნა, რომლითაც ერთეულოვანი წრის ფართობის მქონე კვადრატის აგებას შევძლებდით ფარგლისა და სახაზავის გამოყენებით. სამაგიეროდ შეიძლება ასეთი ფართობის მქონე კვადრატს ნებისმიერი სიზუსტით მივუახლოვდეთ - ამოცანის ოდნავი შეცვლისას იგი ამოხსნადი ხდება: მოცემულია წრე და რაიმე დადებითი რიცხვი (სიზუსტე). ფარგლისა და სახაზავის გამოყენებით ააგეთ ისეთი კვადრატი, რომლის ფართობიც მოცემული სიზუსტით იქნება მოცემული წრის ფართობის სიახლოვეში.
მეორე მაგალითად შეიძლება ჰილბერტის X პრობლემის მოყვანა: მოცემულია ნებისმიერი დიოფანტეს პოლინომიური განტოლება მთელი კოეფიციენტებით: $a_n \cdot x_n^n + \cdots + a_1 \cdot x_1^1 + a_0 = 0$. შეიმუშავეთ მეთოდი (ანუ ალგორითმი), რომელიც ნებისმიერი ასეთი პოლინომისათვის გაგვცემდა პასუხს (კი ან არა) შეკითხვაზე, აქვს თუ არა ამ განტოლებას მხოლოდ რაციონალური ფესვები? როგორც აღმოჩნდა, არც ამ ამოცანას აქვს ალგორითმული ამოხსნა. აღსანიშნავია, რომ ეს ამოცანები ალგორითმის განსაზღვრებაზე ადრე ჩამოყალიბდა და წმინდა მათემატიკური ხერხებით იქნა გადაჭრილი, თუმცა მათ ინფორმატიკაში ძალიან მნიშვნელოვანი როლი ითამაშეს.

პირველი ამოცანა, რომლის ამოუხსნადობაც ალგორითმულ მეთოდებზე დაყრდნობით დამტკიცდა, ე.წ. შეჩერების ამოცანაა: ნებისმიერი ალგორითმისათვის (პროგრამისთვის) და მისი მონაცემისათვის დაადგინეთ, შეჩერდება თუ ჩაიციკლება ეს ალგორითმი მოცემულ მონაცემზე. ამის კონკრეტული მაგალითია სამი პროგრამა, რომელთაგან პირველი რიგ-რიგობით გადაამოწმებს ნატურალურ რიცხვებს (ანუ ნებისმიერ ნატურალურ რიცხვს ოდესმე მაინც მიაღწევს) და შეჩერდება მაშინ და მხოლოდ მაშინ, თუკი ისეთ ლუწ რიცხვს აღმოაჩენს, რომელიც ორი სხვა ლუწი რიცხვის ჯამს წარმოადგენს. მეორე ალგორითმი რიგ-რიგობით გადაამოწმებს ნატურალურ რიცხვებს და შეჩერდება მაშინ და მხოლოდ მაშინ, თუ ისეთ კენტ რიცხვს აღმოაჩენს, რომელიც ორი სხვა კენტი რიცხვის ჯამს წარმოადგენს. ცხადია, რომ პირველ ალგორითმზეც და მეორეზეც წინასწარ შეიძლება იმის თქმა, შეჩერდება თუ არა იგი.
მაგრამ განვიხილოთ ალგორითმი, რომელიც რიგ-რიგობით გადაამოწმებს ლუწ რიცხვებს და შეჩერდება მაშინ და მხოლოდ მაშინ, თუ განხილული რიცხვი მაქსიმუმ ორი მარტივი რიცხვის ჯამისაგან არ შედგება. ეს უკანასკნელი ამოცანა თანამედროვე მათემატიკის ერთ-ერთი უდიდესი ღია პრობლემაა (გოლდბახის ჰიპოთეზა) და,

ცხადია, ჯერ-ჯერობით წინასწარ ვერ ვიტყვით, შეჩერდება თუ არა ზემოთ მოყვანილი ალგორითმი. ამრიგად, ზოგი ალგორითმისთვის დაბეჯითებით შეიძლება იმის თქმა, შეჩერდება თუ არა იგი, ზოგისთვის კი -- არა. მაგრამ აქ მთავარი ისაა, არსებობს თუ არა რაღაც ერთი ზოგადი მეთოდი, რომელიც ყველა ალგორითმისთვის რაიმე პასუხს გაგვცემს? როგორც აღმოჩნდა, ასეთი ზოგადი მეთოდი (ალგორითმი) ვერ იარსებებს (ამის დამტკიცება ტიურინგის მანქანებით ხდება და შემდგომში იქნება მოცემული).
ეს ფაქტი არ ნიშნავს აუცილებლად იმას, რომ რაღაცა ამოცანებისთვის ვერ ვიტყვით, ჩაიციკლება თუ არა მათი ალგორითმები (თუმცა ასეთი ამოცანების რიცხვი უსასრულოა). უსასრულოდ ბევრი ამოცანისათვის დაგვჭირდება ახალი მეთოდების ძიება, რომ ამ შეკითხვაზე პასუხი გავცეთ. შეჩერების ამოცანა ალგორითმულად ამოხსნადი რომ ყოფილიყო, მათემატიკური ლოგიკისა და ინფორმატიკის ერთ-ერთი უმნიშვნელოვანესი განხრა -- თეორემათა ავტომატური მტკიცება -- ადვილად გადაიჭრებოდა, ახლა კი საჭიროა თეორემათა კლასიფიკაცია და ყოველი ცალკეული კლასისათვის ავტომატური მტკიცებების მეთოდების ძიება.

რეალურ დროში ამოხსნადი ამოცანების მაგალითებია:

- ნატურალური რიცხვის სიმარტივის ტესტი: მოცემულია ნატურალური რიცხვი. დაადგინეთ, მარტივია თუ არა იგი.
- ეილერის ციკლი გრაფში: მოცემულია გრაფი. დაადგინეთ, შეიძლება თუ არა შემოვუაროთ ამ გრაფს ისე, რომ მის ყველა წიბოზე გავიაროთ ერთხელ და მხოლოდ ერთხელ? (შეიძლება თუ არა მოცემული ფიგურა შემოვხაზოთ ხელის ერთი მოსმით?)
- პლანარულობის ტესტი: მოცემულია გრაფი. არის თუ არა იგი პლანარული? და თუ არის, როგორ შეიძლება მისი პლანარულად დახაზვა?

ისეთი ამოცანების მაგალითებად, რომელთათვისაც რეალურ დროში ამოხსნადი ალგორითმი ცნობილი არაა (მაგრამ არც ისაა დამტკიცებული, რომ მათთვის ასეთი ალგორითმი არ არსებობს), შეიძლება მოვიყვანოთ:

- ფაქტორიზაცია: მოცემულია ნატურალური რიცხვი. დაშალეთ იგი მარტივ მამრავლებად.
- ჰამილტონის გზა: მოცემულია გრაფი. შეიძლება თუ არა მას ისე შემოვუაროთ, რომ მის ყველა კვანძში შევიდეთ ერთხელ და მხოლოდ ერთხელ?
- ბულის ფუნქციათა ჭეშმარიტება: მოცემულია ბულის ალგებრის n ცვლადზე დამოკიდებული ფუნქცია, რომელიც ჩაწერილია კონიუნქციური ნორმალური ფორმით. არსებობს თუ არა მისი ცვლადების ისეთი მნიშვნელობები, რომ ეს ფუნქცია იყოს ჭეშმარიტი?

აღსანიშნავია, რომ ფაქტორიზაციის გარდა ორივე ამოცანისთვის დამტკიცებულია, რომ იგი NP- *სრულია*, ანუ ჩვენს გარშემო არსებული თითქმის ყველა ამოცანა ამ ამოცანებზე დაიყვანება - ნებისმიერი ამოცანა შეიძლება ისე გარდავქმნათ (ჩავწეროთ ბულის ფუნქციისა ან გრაფის სახით), რომ მიღებული ბულის ფუნქციათა ჭეშმარიტების ან ჰამილტონის ციკლის ამოცანის პასუხი საწყისი ამოცანის პასუხს მოგვცემს (ასეთი გარდაქმნის მაგალითებს შემდგომში განვიხილავთ). „NP-სრული" ამოცანისათვის „რეალურ დროში" ამონახსნის პოვნა ავტომატურად იმას ნიშნავს, რომ თითქმის ყველა სხვა ამოცანაც რეალურ დროში ამოიხსნება: სხვა ამოცანას მასზე დავიყვანთ და შემდეგ ეფექტურად ამოვხსნით. ეს არის ერთ-ერთი მიზეზი იმისა, თუ რატომ ექცევა ამ ამოცანებს ამდენი ყურადღება - იმის გარდა, რომ ეს ამოცანები ძალიან ხშირად იჩენენ თავს ყოველდღიურ ცხოვრებაში.
თეორიული ინფორმატიკა და, კერძოდ, სირთულის თეორია ამოცანების კლასიფიკაციითა და ეფექტურად გადაჭრის გზების პოვნითაა დაკავებული: ყოველი ახალი ამოცანის წამოჭრისას უნდა დავადგინოთ, ხომ არ არის იგი ამოუხსნელი? და შემდეგ, ხომ არ არის იგი NP-სრული? თუ ერთ-ერთი მაინც ჭეშმარიტია, მაშინ უნდა ვეძიოთ (სხვადასხვა მეთოდებზე დაყრდნობით) ამ სირთულეების გვერდის ავლის გზები.
მნიშვნელოვანია შემდეგი გარემოება: ფარგლითა და სახაზავით კონკრეტული გეომეტრიული ფიგურების აგების ამოცანა გადაუჭრელია მხოლოდ გარკვეული მეთოდების გამოყენებით -- თუ ჩვენ საკითხს ისე დავსვამთ, შეიძლება თუ არა იგივე ფიგურების რაიმე მეთოდით აგება, მაშინ პასუხი დადებითი იქნება. დანარჩენი ამოცანები კი ზოგადად გადაუჭრელია -- არ არსებობს რაიმე მეთოდი, რომელიც ამ შეკითხვებზე პასუხს გაგვცემდა. ეს კი იმას ნიშნავს, რომ ამოცანის ჩამოყალიბებისას მნიშვნელოვან როლს თვით რესურსების შერჩევა თამაშობს.

1.2 ანბანი და ენა

ინფორმატიკაში ცენტრალურ როლს რაღაცა ანბანით შექმნილი ენა თამაშობს. როგორც აღმოჩნდა, ნებისმიერი ამოცანა შეგვიძლია ისე ჩამოვაყალიბოთ, რომ მისი პასუხი იყოს ტოლფასი შემდეგ შეკითხვაზე გაცემული პასუხისა: სრულდება თუ არა რაღაცა გარკვეული პირობა? მაგალითად, თუ გვაქვს ზურგჩანთის ამოცანა ელემენტებით $X_1, ..., X_n$ და მაქსიმალური ტევადობით W, იგი შეიძლება დავსვათ შემდეგნაირად: მოცემულია ნატურალური

რიცხვები $X_1, ..., X_n, V$. მოიძებნება თუ არა X მიმდევრობაში ისეთი ქვემიმდევრობა, რომლის ელემენტების ჯამი ზუსტად ტოლია V ? ამ ტიპის კი - არა პასუხიანი ამოცანებით ნებისმიერ ამოცანაზე პასუხის გაცემა შეიძლება. ზემოთ ჩამოყალიბებულ ზურგჩანთის ამოცანაზე პასუხს მივიღებთ მას შემდეგ, რაც მეორე ამოცანაზე გავცემთ პასუხს, როდესაც $V = W, V = W - 1, ..., V = 1$ და პირველივე დადებით პასუხს ავიღებთ.
ანბანად შეგვიძლია ნებისმიერი სასრული სიმრავლე მივიჩნიოთ, მაგალითად ქართული ასოებით შემდგარი ანბანი, ან ქართულ ანბანს დამატებული სასვენი ნიშნები, ან ლათინური ანბანი და ა.შ. ინფორმატიკაში მნიშვნელოვან როლს თამაშობს ორობითი ანბანი $\mathbb{B} = \{0, 1\}$. ანბანით შექმნილი სიტყვა ამ ანბანის ელემენტების მიმდევრობაა. მაგალითად, სიტყვა „ინფორმატიკა" ქართული ანბანით შექმნილი სიტყვაა, „სირთულის თეორია" კი ქართულ ანბანს დამატებული ერთი ელემენტით -- ხარვეზით (ცარიელი სიმბოლოთი) შექმნილი ანბანის სიტყვაა. ორობით ანბანზე შექმნილი სიტყვის მაგალითია 011101010. სიტყვა შეიძლება იყოს სასრული ან უსასრულო ზომის, ან ერთი ელემენტისაგან შემდგარი. მაგალითად, a ლათინური ანბანის ერთი ასო ან ამ ანბანზე შექმნილი ერთ სიმბოლოიანი სიტყვა შეიძლება იყოს. სიტყვა შეიძლება ცარიელიც იყოს, ანუ 0 ელემენტისაგან შედგებოდეს. ამ შემთხვევაში იტყვიან, რომ ეს ცარიელი სიტყვაა. ყოველ სიტყვას აქვს თავისი ზომა, რაც მასში შემავალი ელემენტების რაოდენობის ტოლია. მაგალითად, |ინფორმატიკა| = 11 (სიტყვის სიგრძის აღსანიშნავად მას მოდულის ნიშანს უკეთებენ). თუ ϵ ცარიელი სიტყვაა, მაშინ $|\epsilon| = 0$. თუ მოცემულია რაღაცა ანბანი A, მაშინ $A^n = \{w : |w| = n\}$, ხოლო A^* ამ ანბანზე შექმნილი ყველა სასრული ზომის სიტყვათა სიმრავლეა. თუ w რაღაცა სიტყვაა, მაშინ $w(i)$ ამ სიტყვის i-ურ პოზიციაზე მდგარი ასოა, ხოლო w^R მისი შებრუნებული სიტყვაა. w, v სიტყვებისთვის $w \circ v$ ამ სიტყვების შერწყმაა. მაგალითად, ქვე $\circ$ სიმრავლე = ქვესიმრავლე და ქვესიმრავლე(8) = ა. გარდა ამისა, $w^0 = \epsilon$, $w^{i+1} = w^i \circ w$. მაგალითად, თუ $w =$ ცხელ, მაშინ w^2 ○ი = ცხელცხელი, ხოლო (ცხელცხელი)^R =ილეხცლეხც.

ადვილი შესამჩნევია, რომ არსებობს უსასრულო ენები, ანუ ისეთი ენები, რომელთა სიტყვების რაოდენობა უსასრულოა. გარდა ამისა, ყველა შესაძლო ენათა სიმრავლე არათვლადია (როგორც აღმოჩნდა, ყოველ ენას ერთი ამოცანა მაინც შეესაბამება, ხოლო ყველა შესაძლო ალგორითმის სიმრავლე თვლადია, ესე იგი, უნდა არსებობდეს ისეთი ამოცანებიც, რომელთაც ალგორითმული ამოხსნა არ აქვთ). ამიტომ ინფორმატიკის ერთ-ერთი ძირითადი ამოცანაა ენათა სასრული რესურსებით აღწერა. მნიშვნელოვანია იმის აღნიშვნაც, რომ ენის სასრული აღწერა თვითონ სასრული სიტყვა უნდა იყოს, რომელიც რაღაცა Σ ანბანზეა შექმნილი. გარდა ამისა, ორ სხვადასხვა ენას ორი სხვადასხვა აღწერა უნდა ჰქონდეს. მნიშვნლვანია ერთი ფაქტიც: შესაძლო სასრული აღწერილობების სიმრავლე თვლადია, რადგან Σ^* თვლადია. მაგრამ ყველა ენის სიმრავლე არათვლადია. ამიტომ ყველა ენას სასრული აღწერილობა ვერ ექნება და ჩვენ უნდა ვეძებოთ ასეთი აღწერილობები რაც შეიძლება მეტი საინტერესო და მნიშვნელოვანი ენისათვის. მაგალითისათვის განვიხილოთ შემდეგი ენა:
$L = \{w : w$ სიტყვა ბინარულ ანბანზეა შექმნილი და შეიცავს 2 ან 3 ერთიანს, რომელთაგან პირველი ორი მიყოლებით არ გვხვდება$\}$.
ამ ენის სასრული აღწერილობა შემდეგია: $L = 0^*10^*010^*(10^* \vee e)$.
სიტყვიერად ეს ასე ჩაიწერება: ენა შედგება ისეთი სიტყვებისაგან, რომლებშიც დასაწყისში შეიძლება იყოს რამოდენიმე 0, შემდეგ აუცილებლად მოყვება 1, შემდეგ შეიძლება იყოს რამოდენიმე 0, შემდეგ აუცილებლად მოყვება 01, შემდეგ შეიძლება იყოს რამოდენიმე 0, შემდეგ შეიძლება იყოს 1 და რამოდენიმე ნული (ან არაფერი). აღწერის ბოლოს $(10^* \vee e)$ იმისათვისაა საჭირო, რომ სიტყვის ბოლოს იყოს ან 10^* ან არაფერი.
ასეთი სახის გამოსახულებებს რეგულარული ეწოდება. თუ ენა შეიძლება რეგულარული გამოსახულებით აღიწეროს, მაშინ მას რეგულარული ენა ეწოდება. თუ ენა რეგულარულია, მაშინ იგი უსასრულოდ ბევრი სხვადასხვა რეგულარული გამოსახულებით შეიძლება აღიწეროს.
როგორც აღვნიშნეთ, უნდა არსებობდეს ენები, რომლებიც რეგულარული არაა. ასეთი ენის მაგალითია $\{0^n 1^n : n \geq 0\}$. ეს ფაქტი მკაცრადაა დამტკიცებული ე.წ. პამპინგ ლემის (Pumping Lemma) დახმარებით, მაგრამ აქ იმის ვარაუდი შეიძლება, რომ ეს ენა იმიტომ ვერ აღიწერება სასრული რესურსებით, რომ n რაგინდ დიდი შეიძლება იყოს (სასრული მეხსიერების გადავსება). ასეთი მარტივი ენის აღწერის შეუძლებლობა იმაზე მიგვითითებს, რომ ამდაგვარი აღწერილობა არაა დამაკმაყოფილებელი გამოთვლების ზოგადი თეორიისათვის, რაც ახალი, უფრო ფართო კლასის შექმნას მოითხოვს. ასეთია რეკურსიულ ენათა კლასი, რომელიც ტიურინგის მანქანებზე დაყრდნობით იქნა შემუშავებული.
როგორც ზემოთ აღვნიშნეთ, ნებისმიერი ამოცანა შეიძლება ისე გარდავქმნათ, რომ იგი გადაიქცეს ე.წ. კი-არა ამოცანად: გარკვეულ მონაცემებზე პასუხი იყოს კი და გარკვეულზე -- არა. თუ ჩვენ კონკრეტულ ენად მხოლოდ იმ მონაცემთა სიმრავლეს ავიღებთ, რომელზედაც ამოცანის პასუხია კი, მაშინ ადვილი სანახავია, რომ ეს ტოლფასია ამოცანისა, ვიპოვნოთ პასუხი შეკითხვაზე: არის თუ არა რაღაცა სიტყვა კონკრეტული ენის ელემენტი? მაგალითად, თუ მოცემულ გრაფში ჰამილტონის გზის არსებობის დადგენა გვინდა, შეგვიძლია შევქმნათ ისეთი სიმრავლე HG, რომელიც *ყველა* გრაფის აღწერილობისგან შედგება, რომელიც ჰამილტონის გზას შეიცავს და არც ერთ ისეთს, რომელიც ასეთ გზას არ შეიცავს. აქედან გამომდინარე, თუ მოცემულია გრაფი G, შევქმნით მის აღმწერ სიტყვას w_G და $w_G \in HG$ შეკითხვაზე პასუხი G გრაფში ჰამილტონის გზის არსებობის პასუხს დაემთხვევა. იმისათვის, რომ პასუხი გავცეთ რომელიმე ენაში სიტყვის არსებობის შეკითხვას, საჭიროა ე.წ.

ენის ამომცნობი დანადგარი, ანუ გარკვეული ალგორითმი. როგორც აღმოჩნდა, ზოგადი შემთხვევისათვის ასეთი ალგორითმი არ არსებობს: არ შეიძლება ისეთი ზოგადი ალგორითმის შექმნა, რომელიც ნებისმიერი მოცემული (უსასრულო) ენისათვის და კონკრეტული სიტყვისათვის დაადგენს, არის თუ არა ეს სიტყვა ამ ენის ელემენტი (ეს ე.წ. სიტყვის ამოცანა ჯგუფთა თეორიიდან მოდის და მისივე მეთოდებით დამტკიცდა, რაც ნათლად გვიჩვენებს მათემატიკის მნიშვნელობას ინფორმატიკაში). ამიტომ ამოუხსნადი ამოცანებისათვის უნდა ვეძიოთ მინიმალური ცვლილებები, რაც მათ ამოხსნადად გადააქცევს.
ინფორმატიკაში ცენტრალურ როლს ე.წ. "ჩერჩის თეზისი" თამაშობს: თუ რაიმე ამოცანა ტიურინგის მანქანის საშუალებით არ ამოიხსნება, მაშინ იგი არც ერთი სხვა გამომთვლელით არ ამოიხსნება (ტიურინგის მანქანა და ზოგადად ალგორითმები ერთი და იგივე ცნებაა). სხვა სიტყვებით რომ ვთქვათ, ნებისმიერი ფუნქცია, რომელიც რაიმენაირად გამოითვლება, შეიძლება ტიურინგის მანქანითაც გამოვითვალოთ. ეს თეზისი ფუნდამენტურია მთელი ინფორმატიკისათვის, თუმცა მისი დამტკიცება შეუძლებელია, რადგან არაა მკაცრად განსაზღვრული, რას ნიშნავს "რაიმენაირად გამოთვლილი ფუნქცია". ჯერ-ჯერობით ისეთი ფუნქცია არ მოძებნილა, რომელიც ტიურინგის მანქანით არ გამოითვლება, მაგრამ რაიმე სხვა მეთოდით -- კი.

1.3 ერთი ამოცანის მეორეზე დაყვანა

განვიხილოთ შემდეგი ორი ამოცანა:

- იზოლირებული სიმრავლე გრაფში მოცემულია გრაფი $G = (V, E)$ და ნატურალური რიცხვი k. მოიძებნება თუ არა კვანძების ქვესიმრავლე E' ისეთი, რომ $|E'| > k$ და ამ სიმრავლეში კვანძების არც ერთი წყვილი წიბოთი შეერთებული არ იყოს?
- სრული სიმრავლე გრაფში მოცემულია გრაფი $G = (V, E)$ და ნატურალური რიცხვი k. მოიძებნება თუ არა კვანძების ქვესიმრავლე E' ისეთი, რომ $|E'| > k$ და ამ სიმრავლეში კვანძების ყველა წყვილი წიბოთი შეერთებული იყოს?

თუ მოცემულია იზოლირებული სიმრავლის ამოცანა, მისი გრაფის გარდაქმნა შეიძლება შემდეგნაირად: კვანძების სიმრავლე იგივე რჩება. თუ საწყის გრაფში ორი წვერო წიბოთი შეერთებული არაა, გარდაქმნილ გრაფში ეს ორი წვერო უნდა შეერთდეს და პირიქით: თუ საწყის გრაფში ორი წვერო წიბოთია შეერთებული, მაშინ გარდაქმნილ გრაფში მათ შორის წიბო არ იარსებებს. ადვილი სანახავია, რომ ასეთ გარდაქმნილ გრაფში სრული სიმრავლის ამოცანა პასუხს გაგვცემს იზოლირებული სიმრავლის ამოცანაზე, ანუ პირველი ამოცანის მეორეზე დაყვანა შეიძლება.
როგორც აღმოჩნდა, ეს ორივე ამოცანა (იმის და მიუხედავად, რომ ერთი შეხედვით ეს ორივე ამოცანა მარტივად გამოიყურება) თავიანთი კლასისათვის ე.წ. სრული ამოცანებია, ანუ ამ კლასის ყველა დანარჩენი ამოცანა მათზე დაიყვანება გარკვეული გარდაქმნების გზით. საკმარისია იმის ჩვენება, რომ ეს ამოცანები რაიმე სხვა ამოცანაზე დაიყვანება, რომ ის ამოცანაც სრული ხდება: ჯერ დავიყვანთ ნებისმიერ ამოცანას სრული სიმრავლის ამოცანაზე და მერე სრული სიმრავლის ამოცანას ამ უკანასკნელზე. აქ გასათვალისწინებელია ის მომენტი, რომ დაყვანის პროცესში (ალგორითმში) ბიჯების რაოდენობა პოლინომიური ფუნქციით იყოს შემოსაზღვრული. რადგან ორი პოლინომიური ფუნქციის კომპოზიციაც თვითონ პოლინომიურია, შესაძლებელია დაყვანის რაგინდ გრძელი ჯაჭვის აგება შედეგად მაინც ისეთ ალგორითმს მივიღებთ, რომლის ბიჯების რაოდენობაც ზემოდან პოლინომიური ფუნქციით იქნება შემოსაზღვრული. აქედან გამომდინარეობს, რომ თუ შესაძლებელი იქნება ასეთი სრული ამოცანების პოლინომიურ დროში გადაჭრა, ნებისმიერ სხვა ამოცანასაც გადავჭრიდით პოლინომიურ დროში. ამიტომ ამ ამოცანებისათვის ეფექტური ალგორითმების პოვნა ან იმის დამტკიცება, რომ ასეთი ალგორითმი არ არსებობს (ამოცანის ექსპონენციური ქვედა ზღვრის ჩვენება), ცენტრალური საკითხია ინფორმატიკაში. იმისათვის, რომ რაიმე ამოცანის სისრულე დავამტკიცოთ, რაიმე სრული ამოცანა უნდა დავიყვანოთ მასზე. ცხადია, რომ უნდა არსებობდეს ამოცანა, რომლის სისრულეც სულ პირველად რაღაცა სხვა მეთოდებით დამტკიცდა. ასეთია ბულის ფუნქციათა ჭეშმარიტების ამოცანა, რომელიც ზემოთ იყო ნახსენები.

გადაფარვის ამოცანა: მოცემულია გრაფი $G = (V, E)$ და ნატურალური რიცხვი k. ვიტყვით, რომ გრაფის წვერო გადაფარავს იმ წიბოებს, რომლებიც ამ წვეროსთან არიან დაკავშირებული. თუ მოცემულია წვეროების რაღაცა სიმრავლე, იტყვიან, რომ იგი მთელ გრაფს გადაფარავს, თუ მისი ნებისმიერი წიბო მოცემულ სიმრავლეში შემავალი ერთი წვეროთი მაინც გადაიფარება. შეიძლება თუ არა მოცემული გრაფის k წვეროიანი სიმრავლის გამოყოფა, რომელიც ამ გრაფს გადაფარავს?

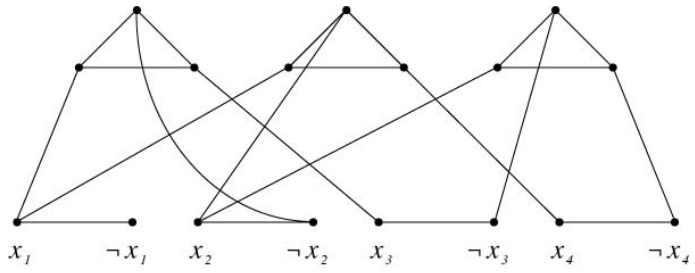
იმის საჩვენებლად, რომ ეს ამოცანა სრულია, უნდა დავამტკიცოთ, რომ რომელიმე სრული ამოცანა, მაგალითად ბულის ფუნქციათა ჭეშმარიტების ამოცანა, მასზე დაიყვანება: ამ ბოლო ამოცანის ელემენტები - ბულის ფუნქციის k ცვლადი, შეერთებული კონიუნქციითა და დიზიუნქციით, გარდავქმნათ გრაფად, რომელიც m წვეროთი

გადაიფარება მაშინ და მხოლოდ მაშინ, თუ ეს ფუნქცია ოდესმე ჭეშმარიტი გახდება (აქ m დამოკიდებულია k -ზე).

კერძო მაგალითი: მოცემულია ფუნქცია, რომელიც ჩაწერილია კონიუნქციური ნორმალური ფორმით და შედგება 4 ცვლადისა და 3 დიზიუნქციური ჩანაწერისაგან (თითოეულ ასეთ ჩანაწერში შედის ზუსტად 3 ცვლადი -- დამტკიცებულია, რომ ასეთი სახის ფუნქციებისათვისაც ჭეშმარიტების ამოცანა NP სრულია):

$$F(x_1, x_2, x_3, x_4) = (x_1 \vee \neg x_2 \vee x_3)(x_1 \vee x2 \vee x4)(x_2 \vee \neg x_3 \vee \neg x_4).$$

ყოველ ცვლადს შევუსაბამოთ ერთი კვანძი გრაფში, ხოლო ყოველ დიზიუნქციურ ჩანაწერს შევუსაბამოთ სამი ერთმანეთთან წიბოთი შეერთებული კვანძი (იხ. ნახ. **??**). თუ ჩანაწერში შედის რომელიღაცა ცვლადი x_i ან $\neg x_i$, მისი შესაბამისი კვანძი ჩანაწერის ერთ-ერთ კვანძს უერთდება წიბოთი. ადვილი საჩვენებელია, რომ k ცვლადიანი და c დიზიუნქციურ ჩანაწერიანი მოცემული ფუნქციის ასეთი სახით გარდაქმნის შედეგად მიღებული გრაფი გადაიფარება $k + 2c$ კვანძით მაშინ და მხოლოდ მაშინ, თუ ეს ფუნქცია შეიძლება ჭეშმარიტი გახდეს ცვლადების გარკვეული მნიშვნელობებისათვის (თუ არსებობს ცვლადების ასეთი მნიშვნელობები, მაშინ უნდა შეირჩეს მათი შესაბამისი კვანძები).



ნახ. 1: რედუქციის სქემა

თუ ერთი ამოცანა მეორეზე დაიყვანება, მაშინ ეს მეორე ამოცანა ამ პირველზე უფრო რთული უნდა იყოს, ანუ მეორე ამოცანის გადასაჭრელად საჭირო ოპტიმალური ალგორითმის ბიჯების რაოდენობა არაა ნაკლები პირველი ამოცანის გადასაჭრელად საჭირო ოპტიმალური ალგორითმის ბიჯების რაოდენობაზე.

1.4 ამოცანათა სირთულის კლასები

რადგან ერთსა და იმავე ამოცანას ბევრი სხვადასხვა ალგორითმული ამოხსნა აქვს, შეიძლება წამოიჭრას ბუნებრივი შეკითხვა: როდის ჯობია ერთი ალგორითმი მეორეს? ამ შეკითხვაზე ბუნებრივი პასუხია: როდესაც ერთი ალგორითმი ნაკლებ რესურსს ხარჯავს, ვიდრე მეორე -- ნაკლებ დროსა და ადგილს (მეხსიერების უჯრედს).
არსებობს ალგორითმთა ორი ტიპი: პარალელური და მიმდევრობითი ალგორითმები. ძირითადად ხმარობენ პარალელურ ალგორითმებს -- კომპიუტერულ მიკროსქემებში, გამომთვლელ დანადგარებში, პარალელურ პროცესორიან მანქანებში და ა.შ. ბუნებრივად ისმის შეკითხვა: რამდენად უფრო ეფექტურია პარალელური ალგორითმი მიმდევრობითთან შედარებით? აღმოჩნდა, რომ ალგორითმების „გაპარალელება" გარკვეულ უპირატესობას იძლევა: მაგალითად, n რიცხვის დახარისხების ამოცანა მიმდევრობით ალგორითმში $O(n \log n)$ ბიჯს ანდომებს, ხოლო პარალელურში კი -- $O(\log n)$. ორი n ბიტიანი რიცხვის მიმატების ამოცანა მიმდევრობით ალგორითმში $O(n)$ ბიჯს ანდომებს, ხოლო პარალელურში კი -- $O(\log n)$. ანალოგიურადაა საქმე ორი n ბიტიანი რიცხვის გამრავლების ალგორითმში: მიმდევრობით ალგორითმში $O(n \log n \log \log n)$ ბიჯს ანდომებს, ხოლო პარალელურში კი -- $O(\log n)$.
არაა ძნელი იმის ჩვენება, რომ რამდენ მეხსიერებასაც დაიჭერს პარალელური ალგორითმი, ზუსტად იმდენ ბიჯს ანდომებს მისი შესაბამისი მიმდევრობითი ალგორითმი იგივე ამოცანის გადაჭრას. ამიტომაც ალგორითმთა თეორიაში ხშირად პარალელურ ალგორითმებს ეძებენ და მერე მისი დროისა და მეხსიერების მინიმიზაციას ცდილობენ.

რაზეა დამოკიდებული ალგორითმის სისწრაფე? რა თმა უნდა, რაც მეტი მონაცემი იქნება დასამუშავებელი, მით უფრო მეტ დროს (ბიჯს) მოანდომებს ალგორითმი გამოთვლას. მაგლითად, n ნატურალური რიცხვის დახარისხებას $O(\log n)$ დრო ჭირდება. რაც უფრო დიდია n, მით მეტი ბიჯია საჭირო. მაგრამ მნიშვნელობა აქვს აგრეთვე დასახარისხებელ რიცხვთა ზომასაც: n ცალ 1 ბიტიან რიცხვს უფრო სწრაფად დავახარისხებთ, ვიდრე n ცალ

1024 ბიტიან რიცხვს. აქედან გამომდინარე, ალგორითმების ბიჯების რაოდენობა დამოკიდებულია *მონაცემთა საერთო ზომაზე*. n ცალი k ბიტიანი რიცხვის დახარისხებისას მონაცემთა ზომაა $m = n \cdot k$ და ალგორითმის ბიჯების რაოდენობაც იქნება $O(\log m) = O(\log(n \cdot k)) = O(\log n + \log k) = O(\log n)$ (როგორც წესი, მონაცემთა რიცხვი უფრო სწრაფად იზრდება, ვიდრე თითოეული მონაცემის ზომა).
სხვაგვარადაა საქმე ზურგჩანთის ამოცანის ალგორითმისას: მოცემული გვაქვს n ცალი ტვირთი და რაღაცა ნატურალური რიცხვი W (ზურგჩანთაში ჩატეული ტვირთის მაქსიმალური წონა). აქ W ცვლადია, ამიტომ იგი ფაქტიურად ნებისმიერი შეიძლება იყოს. თუ მის აღსაწერად გამოყოფილია k ბიტი, მაშინ მონაცემთა საერთო ზომა უნდა იყოს $(n + 1) \cdot k$ (თითოეული ტვირთის მაქსიმალური წონა და W მაქსიმუმ k ბიტიანი რიცხვი შეიძლება იყოს). ამრიგად, $(n+1) \cdot k$ სიგრძის მონაცემისათვის ალგორითმი ანდომებს $O(n \cdot k \cdot W)$ ბიჯს, რაც, თავის მხრივ, უარეს შემთხვევაში *შეიძლება* იყოს $O(n \cdot k \cdot 2^k)$. ამრიგად, $O(n \cdot k)$ მონაცემის დასამუშავებლად გვჭირდება $O(n \cdot k \cdot 2^k)$ ბიჯი, რაც, მკაცრად რომ ვთქვათ, ექსპონენციური ზრდის რიგისაა, მაგრამ, რადგან პრაქტიკაში k უფრო ნელა იზრდება, ვიდრე n, ეს ექსპონენციური ზრდა შეუმჩნეველია პატარა W-თვის. მაგრამ საკმარისია W რამოდენიმე მილიონის ფარგლებში იყოს, ეს პრობლემა აუცილებლად იჩენს თავს.
ზურგჩანთის ამოცანისაგან განსხვავებით დახარისხების ამოცანაში ყურადღება არ ექცევა დასახარისხებელი რიცხვების მნიშვნელობებს -- ალგორითმის ბიჯების რაოდენობა *მხოლოდ* ამ რიცხვების რაოდენობასა და სიგრძეზა დამოკიდებული, ხოლო ზურგჩანთის ამოცანაში კი ბიჯების რაოდენობაში გადამწყვეტ როლს თვით W თამაშობს, რაც k ბიტის შებთხვევაში $2^k - 1$ შეიძლება იყოს. ამიტომ ზურგჩნთის ამოცანის ალგორითმს „ფსევდოპოლინომიურს" უწოდებენ, რადგან იგი კი არის ექსპონენციური, მაგრამ პრაქტიკაში შემხვედრი შედარებით პატარა რიცხვებისათვის მისი გამოთვლის ბიჯების რაოდენობა თითქმის პოლინომიურად იზრდება.

ინფორმატიკაში ცენტრალურ როლს ამოცანათა ე.წ. *NP* კლასი თამაშობს -- არადეტერმინისტული ტიურინგის მანქანებით პოლინომიურ დროში ამოხსნად ამოცანათა კლასი. აქედან მომდინარეობს შემოკლებაც -- *NP* კლასი -- Nondeterministic Polynomial (ეს მკაცრად შემდგომში იქნება ჩამოყალიბებული).
მაგალითად, ამ კლასში შედის ჰამილტონის ციკლის ამოცანა, გრაფების გადაფარვის ამოცანა, გრაფში იზოლირებულ წერტილთა ამოცანა და მრავალი სხვა. საინტერესოა შემდეგი ფაქტი: თუ მოცემულია გრაფი და მისი წვეროების რაღაცა მიმდევრობა, ადვილი გადასამოწმებელია, შეადგენს თუ არა ეს მიმდევრობა ჰამილტონის ციკლს ამ გრაფისათვის. ასევე ადვილი გადასამოწმებელია ნებისმიერი გრაფისათვის და მისი k წიბოიანი სიმრავლისათვის, შეადგენს თუ არა ეს სიმრავლე იზოლირებულ წვეროთა სიმრავლეს ამ გრაფისთვის, ან გადაფარავს თუ არა ეს სიმრავლე მთლიან გრაფს? ზოგადად, თუ მოცემულია რაიმე ამოცანა და შემოგვთავაზეს მისი სავარაუდო ამონახსნი, ადვილად (ანუ პოლინომიური ალგორითმით) შეიძლება იმის შემოწმება, მართლაა თუ არა ეს სავარაუდო შემოთავაზებული ამონახსნი ამ ამოცანის ამონახსნი?
ამოცანათა ამ კლასში შედის აგრეთვე ისეთი ამოცანები, რომელთა ამოხსნაც დეტერმინისტული მანქანებით შეიძლება პოლინომიურ დროში (ანუ ისეთი ალგორითმებით, რომელსაც ჩვენ ვართ შეჩვეული), რადგან მათთვის არა თუ შემოთავაზებული სავარაუდო ამონახსნის გადამოწმება შეიძლება პოლინომიურ დროში, არამედ თვით ამ ამონახსნის *პოვნაც* კი.
როგორც ზემოთ აღვნიშნეთ, არსებობს ისეთი ამოცანები, რომელზედაც რაღაცა კლასის ყველა ამოცანა პოლინომიურ დროში დაიყვანება. თუ $\mathcal{A}$ კლასში შემავალი ყველა ამოცანა რომელიღაცა B ამოცანაზე დაიყვანება, მაშინ იტყვიან, რომ B ამოცანა $\mathcal{A}$-რთულია. თუ ამასთან ერთად $B \in \mathcal{A}$, მაშინ იტყვიან, რომ ამოცანა B $\mathcal{A}$-სრულია. როგორც აღმოჩნდა, ჰამილტონის ციკლის ამოცანა, გრაფთა გადაფარვის ამოცანა, ბულის ფუნქციათა ჭეშმარიტების ამოცანა, ზურგჩანთის ამოცანა და მრავალი სხვა *NP*-სრულია: ეს ამოცანები შედიან *NP* სიმრავლეში და ამ კლასში შემავალი *ყველა* ამოცანა ამ ამოცანებზე დაიყვანება პოლინომიურ დროში.
ამრიგად, თუ ვინმე რომელიმე *NP*-სრული ამოცანისათვის პოლინომიურ ალგორითმს დაწერს, მაშინ ასეთი სწრაფი ალგორითმი ავტომატურად აღმოჩნდება ამ კლასის *ყველა* სხვა ამოცანისთვისაც. დღეისათვის ინფორმატიკის ყველაზე დიდი პრობლემა შემდეგნაირად შეიძლება ჩამოყალიბდეს:

$$P = NP \ ?$$

თუ ერთ-ერთი *NP*-სრული ამოცანისთვის მაინც მოიძებნება პოლინომიური ალგორითმი, მაშინ ასეთი ალგორითმი ამ კლასის ყველა დანარჩენი ამოცანისთვისაც მოიძებნება და $P = NP$. თუ დამტკიცდა, რომ ერთი *NP*-სრული ამოცანისთვის მაინც მისი ქვედა ზღვარია $\Omega(2^n)$, მაშინ დამტკიცდება, რომ *NP* კლასში გვაქვს ისეთი ამოცანა, რომელიც პოლინომიურ დროში არ ამოიხსნება და $P \neq NP$. ეს ამოცანა კლეის მათემატიკურმა ინსტიტუტმა ე.წ. ათასწლეულის პრობლემათა შორის დაასახელა და მის გადაჭრაში დიდი ჯილდოც დააწესა.

1.5 სასრული ავტომატები და ტიურინგის მანქანები

როგორც ზემოთ აღვნიშნეთ, ნებისმიერი ამოცანა შეიძლება ისე გარდავქმნათ, რომ იგი გადაიქცეს ე.წ. კი-არა ამოცანად, კერძოდ არის თუ არა რაღაცა სიტყვა კონკრეტული ენის ელემენტი?

კონკრეტული ენისათვის შეიძლება შეიქმნას რაღაცა „დანადგარი" -- სასრული ავტომატი, რომელიც მოცემული სიტყვისათვის განსაზღვრავს, შედის თუ არა იგი მოცემულ ენაში. ეს სასრული ავტომატი ასო-ასო წაიკითხავს შემავალ სიტყვას და საბოლოოდ მოგვცემს პასუხს -- კი ან არა. რეალურ კომპიუტერებთან მას ის აქვს საერთო, რომ ორივე სასრული ზომის „პროცესორისაგან" შედგება. მაგრამ, როგორც აღმოჩნდა, ასეთი პრიმიტიული ავტომატი ზუსტად იმავე შეკითხვებზე გვცემს პასუხს, როგორებზეც ჩვეულებრივი კომპიუტერი გაგვცემდა! ერთი შეხედვით ეს გასაოცარი ფაქტია, მაგრამ კომპიუტერების რესურსიც -- მეხსიერება, პროცესორის მდგომარეობები და სხვა -- შეზღუდულია. კარგი ფილოსოფიური დისკუსიის შესაძლებლობას იძლევა სამყაროში არსებული რესურსების შეზღუდულობა -- რა იქნებოდა, რესურსები შეზღუდული რომ არ ყოფილიყო? შევძლებდით სხვა ფუნქციების გამოთვლას (ამოცანების გადაჭრას), რასაც ახლა ვერ ვახერხებთ? ამაზე პასუხი ტიურინგის მანქანებით შეიძლება გავცეთ -- სასრული ავტომატების უსასრულო მეხსიერებითა და რამოდენიმე მცირე ფუნქციით გაფართოებული გამომთვლელით.

მათემატიკურად სასრული ავტომატი შემდეგნაირად შეიძლება განვსაზღვროთ:

სასრული ავტომატი არის ხუთეული $(K, \Sigma, \delta, s, F)$, სადაც

K არის სასრული ავტომატის მდგომარეობათა სასრული სიმრავლე;

Σ არის ანბანი;

$\delta : K \times \Sigma \to K$ არის მდგომარეობათა ფუნქცია;

$s \in K$ არის საწყისი მდგომარეობა;

$F \subset K$ არის საბოლოო მდგომარეობათა სიმრავლე.

მუშაობის დაწყებისას სასრული ავტომატი საწყის მდგომარეობას შია. მას შემდეგ, რაც ავტომატი პირველ სიმბოლოს a_0 წაიკითხავს, იგი გადავა იმ მდგომარეობაში, რომელიც $\delta(s, a_0) = k_1$ ფუნქციით განისაზღვრება. მეორე სიმბოლოს a_1 წაკითხვის შემდეგ იგი გადადის იმ მდგომარეობაში, რომელიც განისაზღვრება δ ფუნქციით: $\delta(k_1, a_1) = k_2$. ზოგადად, თუ ავტომატი არის k_i მდგომარეობაში და შემომავალი სიმბოლოა a_i, იგი გადადის მდგომარეობაში $\delta(k_i, a_i) = k_{i+1}$. თუ მთელი სიტყვის წაკითხვის შემდეგ ავტომატი აღმოჩნდა მდგომარეობაში k და $k \in F$, ამბობენ, რომ მანქანამ გასცა პასუხი „კი" და ეს სიტყვა მოცემული ენის ელემენტია (იმ ენისა, რომელსაც განსაზღვრავს ეს ავტომატი). წინააღმდეგ შემთხვევაში ეს სიტყვა ამ ენის ელემენტი არაა. სხვა სიტყვებით რომ ვთქვათ, ზოგიერთი ენისათვის შეიძლება შეიქმნას სასრული ავტომატი, რომელიც ნებისმიერი სიტყვისათვის განსაზღვრავს, შედის თუ არა ეს სიტყვა ამ ენაში.

თუ სასრულმა ავტომატმა $(a_0, a_1, ..., a_i)$ სიმბოლოების წაკითხვის შემდეგ მიაღწია მდგომარეობას k, მაშინ იტყვიან, რომ მისი *კონფიგურაციაა* $(k, a_i...a_1a_0)$. თუ სასრული ავტომატის კონფიგურაციაა $(k_1, a_i...a_1a_0)$ და სიმბოლო a_{i+1} წაკითხვის შემდეგ იგი გადავა კონფიგურაციაში $(k_2, a_{i+1}a_i...a_1a_0)$, მაშინ წერენ: $(k_2, a_{i+1}a_i...a_1a_0) \vdash_M (k_1, a_i...a_1a_0)$. თუ w_1 და w_2 რაიმე სიტყვებია და (k_1, w_1) და (k_2, w_2w_1) სასრული ავტომატის მდგომარეობებია, მაშინ ვწერთ: $(k_1, w_1) \vdash_M^* (k_2, w_2w_1)$ (ერთი მდგომარეობიდან რამოდენიმე სიმბოლოს წაკითხვის შემდეგ რამოდენიმე ბიჯში გადავდივართ მეორე მდგომარეობაში).
იმ ენას, რომელსაც სასრული ავტომატი M განსაზღვრავს (იმ სიტყვათა სიმრავლე, რომელსაც ეს ავტომატი მიიღებს) აღვნიშნავთ $L(M)$.

<u>მაგალითი:</u> $M = (K, \Sigma, \delta, s, F)$, სადაც

$K = \{q_0, q_1\}$
$\Sigma = \{a, b\}$
$s = q_0$
$F = \{q_0\}$

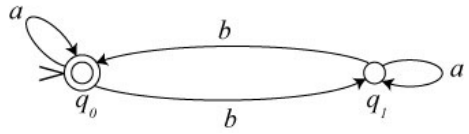
q	σ	$\delta(q, \sigma)$
q_0	a	q_0
q_0	b	q_1
q_1	a	q_1
q_1	b	q_0

ადვილი დასანახია, რომ $L(M) \subset \{a, b\}^*$ ისეთ სიტყვათა სიმრავლეა, რომლებშიც b სიმბოლოების რაოდენობა ლუწია. თუ ავტომატი მიიღებს სიტყვას $(aababa)$, მისი მდგომარეობები იქნება:

$$(q_0, \epsilon) \vdash_M (q_0, a) \vdash_M (q1, ba) \vdash_M (q_1, a) \vdash_M (q_1, aba) \vdash_M (q_0, baba) \vdash_M (q_0, ababa) \vdash_M (q_0, aababa).$$

რადგან ბოლო სიმბოლოს დამუშავების შემდეგ ავტომატი q_0 მდგომარეობაშია, რომელიც საბოლოოა, ავტომატმა ეს სიტყვა მიიღო და, აქედან გამომდინარე, იგი ზემოთ აღწერილი ენის ელემენტია (ლუწი რაოდენობის b სიმბოლოს შეიცავს).

ავტომატების ტაბულარული აღწერა ადამიანებისათვის აღსაქმელად საკმაოდ ძნელია. ამიტომ შემოიღეს ავტომატების დიაგრამებით აღწერის მეთოდი. ზემოთ აღწერილი ავტომატის დიაგრამა მოყვანილია ნახატში 2.

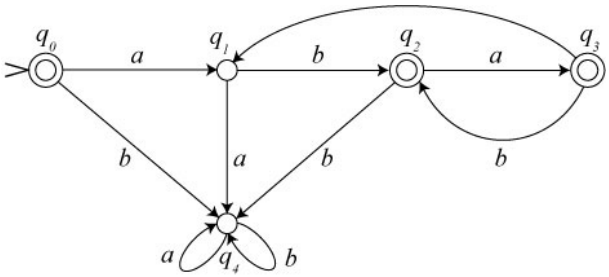


ნახ. 2: სასრული ავტომატის დიაგრამა

ეს იგივე მიმართული წონიანი გრაფია, რომელშიც ყოველ წვეროს ცალსახად ავტომატის ერთი მდგომარეობა შეესაბამება და (q_1, q_2) წიბოს აწერია სიმბოლო a მაშინ და მხოლოდ მაშინ, თუ $\delta(q_1, a) = q_2$. ამას გარდა, საბოლოო მდგომარეობები ორმაგი წრითაა აღნიშნული, ხოლო საწყისი მდგომარეობა კი > სიმბოლოთი.

სასრული ავტომატების აღწერა შეიძლება საკმაოდ რთული იყოს -- ანბანზე, მდგომარეობათა სიმრავლეზე და მდგომარეობათა ფუნქციაზე დამოკიდებული ზომა სწრაფად შეიძლება გაიზარდოს. ქვემოთ მოყვანილია სასრული ავტომატის მაგალითი, რომელიც შემდეგ ენას განსაზღვრავს:

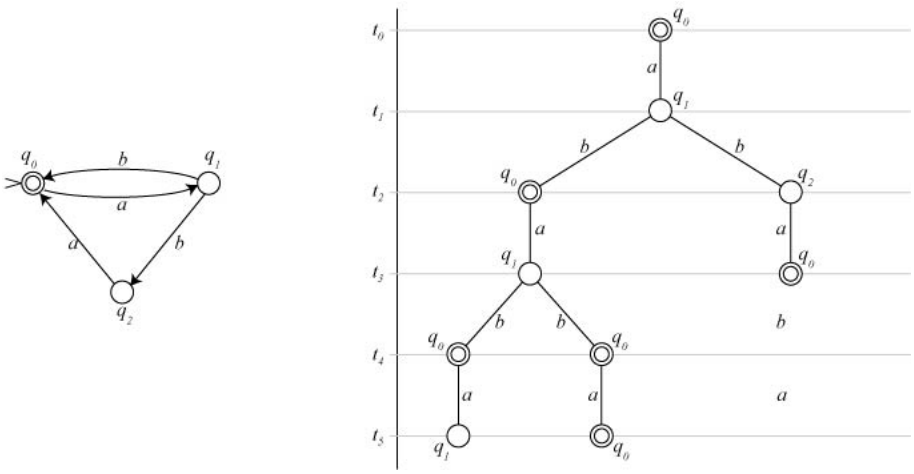
$$L = (ab \cup aba)^*.$$



ნახ. 3: დეტერმინისტული სასრული ავტომატის მაგალითი

აქამდე განხილულ ყველა სასრულ ავტომატს ის თვისება აქვს, რომ ყოველ ბიჯზე კონკრეტული შემომავალი სიმბოლოს წაკითხვისას იგი ერთი მდგომარეობიდან ცალსახად გადადის მეორე მდგომარეობაში მდგომარეობათა ფუნქციის საშუალებით. ასე რომ, ნებისმიერი დეტერმინისტული სასრული ავტომატისათვის ცალსახად შეიძლება იმის თქმა, თუ რა მდგომარეობაში გადავა იგი ამა თუ იმ მდგომარეობიდან გარკვეული სიმბოლოს წაკითხვის შემდეგ. ასეთი სახის შეზღუდვა მოხსნილია ე.წ. „არადეტერმინისტულ" სასრულ ავტომატებში. თუ ასეთი ავტომატი არის მდგომარეობაში q და კითხულობს სიმბოლოს a, მას შეიძლება ჰქონდეს შემდეგი მდგომარეობების არჩევის რამოდენიმე ვარიანტი რაღაცა სიმრავლიდან $\{q_1, q_2, ..., q_l\}$. არჩევის მექანიზმი არადეტერმინისტულია (არ არის ცალსახად განსაზღვრული), რის გამოც ასეთ ავტომატებს არადეტერმინისტულს უწოდებენ. ქვედა ნახაზში მოყვანილია არადეტერმინისტული ავტომატი, რომელიც ზემოთ მოყვანილი დეტერმინისტული სასრული ავტომატის ტოლფასია -- იგივე ენას განსაზღვრავს. ადვილი დასანახია, რომ $L = (ab \cup aba)^*$ ენაში შემავალი ყველა სიტყვისათვის არსებობს რაღაცა გზა საწყისი მდგომარეობიდან საბოლოო მდგომარეობამდე და ყველა სიტყვისათვის, რომელიც ამ ენაში არ შედის, ასეთი გზა არ არსებობს. აქ იჩენს თავს დეტერმინისტულ და არადეტერმინისტულ ავტომატთა ძირითადი განსხვავება: თუ დეტერმინისტულში ერთადერთი ასეთი გზა უნდა არსებობდეს, არადეტერმინისტულში საკმარისია რაიმე გზის არსებობა.

აღსანიშნავია, რომ შემომავალი სიტყვის ასო-ასო წაკითხვისას არადეტერმინისტულ ავტომატში ჩვენ გზას ვერ განვსაზღვრავთ, რადგან, მაგალითად, არ ვიცით, თუ რომელ მდგომარეობაში უნდა გადავიდეთ q_1-დან, თუ წავიკითხეთ სიმბოლოს b -- q_2 თუ q_0 მდგომარეობაში? ეს, ალბათ, იმის ბრალია, რომ ადამიანებიც დეტერმინისტულად



ნახ. 4: არადეტერმინისტული სასრული ავტომატის დიაგრამა

აზროვნებენ და, აქედან გამომდინარე, არადეტერმინისტული ავტომატის თეორიული მოდელის შერჩევის მექანიზმი ჩვენთვის უცნობი დარჩება. მაგრამ მთავარია ის ფაქტი, რომ ჩვენ არადეტერმინისტული ავტომატების სასრული ენით აღწერა შეგვიძლია:

არადეტერმინისტული სასრული ავტომატი $M = (K, \Sigma, s, F, \Delta)$ აღიწერება შემდეგი ხუთეულით:

K არის სასრული ავტომატის მდგომარეობათა სასრული სიმრავლე;

Σ არის ანბანი;

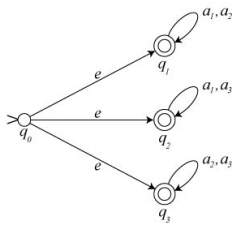
$s \in K$ არის საწყისი მდგომარეობა;

$F \subset K$ არის საბოლოო მდგომარეობათა სიმრავლე;

$\Delta \subset K \times (\Sigma \cup \{e\}) \times K$ არის მდგომარეობათა გადასვლის მიმართება.

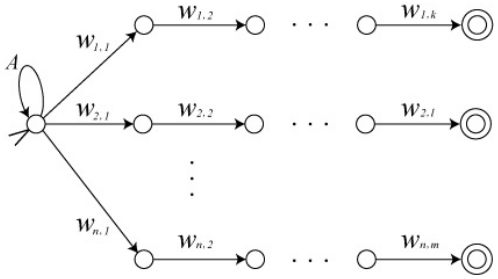
ზემოთ მოყვანილ მაგალითში
$K = \{q_0, q_1, q_2\}$, $\Sigma = \{a, b\}$, $s = q_0$, $F = \{q_0\}$,
$\Delta = \{(q_0, a, q_1), (q_1, b, q_0), (q_1, b, q_2), (q_2, a, q_0)\}$. აქ (q_1, b, q_0), (q_1, b, q_2) ნიშნავს, რომ თუ ავტომატი არის მდგომარეობაში q_1 და წაიკითხავს სიმბოლოს b, იგი გადავა ან q_0 ან q_2 მდგომარეობაში.
ნახ. 4-ში მოყვანილია არადეტერმინისტული სასრული ავტომატის გამოთვლის დიაგრამა შემავალი სიტყვისათვის *"ababa"*. t_0 მომენტში ავტომატის მდგომარეობაა q_0. თუ იგი კითხულობს ასოს a, იგი ცალსახად გადადის მდგომარეობაში q_1, ხოლო თუ წაიკითხავს b, მისთვის გადასასვლელი დიაგრამა არ იარსებებს და გამოთვლის პროცესი შეჩერდება უარყოფითი შედეგით (ავტომატი სიტყვას არ მიიღებს).
ჩვენს მაგალითში ავტომატი კითხულობს სიმბოლოს a და გადადის მდგომარეობაში q_1. შემდეგ ბიჯში, თუ ავტომატი კითხულობს სიმბოლოს a, აქაც არ იარსებებს გადასვლის საშუალება და ავტომატი ჩერდება უარყოფითი შედეგით. თუ წაიკითხავს სიმბოლოს b, არსებობს ორი ვარიანტი: ან გადადის მდგომარეობაში q_1, ან მდგომარეობაში q_2. ეს არის არადეტერმინისტული გადასვლის მომენტი, რომლის დროსაც ჩვენ *პარალელურად* ორივე შემთხვევას განვიხილავთ. შემდგომში ავტომატი ანალოგიურად აგრძელებს მუშაობას.
არადეტერმინისტული ავტომატის გამოთვლის ბიჯების დასათვლელად ჩვენ ზემოთ მოყვანილი ხის სიღრმეს ვითვლით, ხოლო იმის დასადგენად, მიიღებს თუ არა ეს ავტომატი ამა თუ იმ სიტყვას, უნდა დავადგინოთ, არსებობს თუ არა ასეთი ხის ფესვიდან გზა ისეთ ფოთლამდე, რომელიც ავტომატის სასრულ მდგომარეობას აღნიშნავს.
აქ იჩენს თავს არადეტერმინისტულობის ძირითადი უპირატესობა: „გაპარალელების" თვისება. ავტომატი ალტერნატიულ გზებს ერთდროულად განიხილავს და იქიდან ერთ-ერთს ირჩევს.
ჩვენს ზემოთ მოყვანილ მაგალითში, დეტერმინისტული ავტომატისაგან განსხვავებით, q_0 მდგომარეობიდან არ არსებობს b სიმბოლოთი აღნიშნული მიმართული წიბო, რაც იმას ნიშნავს, რომ თუ ამ მდგომარეობაში მყოფი ავტომატი წაიკითხავს b სიმბოლოს, იგი შემავალ სიტყვას არ მიიღებს.
ამას გარდა, არადეტერმინისტული ავტომატი შეიძლება ერთი მდგომარეობიდან მეორეზე „გადახტეს" სიმბოლოს წაკითხვის გარეშე --- ამ შემთხვევაში იტყვიან, რომ შემოვიდა ცარიელი სიმბოლო e.

მაგალითისათვის განვიხილოთ შემდეგი ენა:
მოცემულია ანბანი $\Sigma = \{a_1, ..., a_n\}$. $L = \{w \mid \exists a_i \in \Sigma,\ a_i \notin w\}$. თუ ავიღებთ სამ ელემენტიან ანბანს $\Sigma = \{a_1, a_2, a_3\}$, შესაბამისი ენის განმსაზღვრელი არადეტერმინისტული სასრული ავტომატი ქვედა ნახაზშია ნაჩვენები. მისი ანალოგიური დეტერმინისტული ავტომატი კი საკმაოდ რთულია.



ნახ. 5: არადეტერმინისტული სასრული ავტომატის დიაგრამა

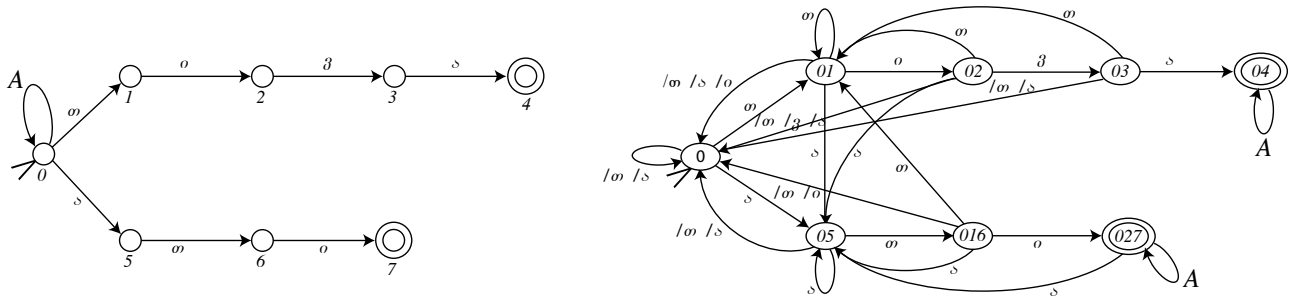
ჭეშმარიტია შემდეგი **თეორემა**: შესაძლებელია ნებისმიერი არადეტერმინისტული სასრული ავტომატის „გადეტერმინისტულება", ანუ ყველა არადეტერმინისტულ ავტომატს ერთი მაინც დეტერმინისტული შეესაბამება, რომელიც იგივე ენას განსაზღვრავს.
ეს უმნიშვნელოვანესი შედეგი იმის მაუწყებელია, რომ თვით ისეთი ძლიერი აპარატი, როგორიცაა არადეტერმინისტულობა, უმთავრეს შედეგზე -- ენათა ამოცნობაზე -- გავლენას ვერ ახდენს. პრობლემა მხოლოდ იმაში მდგომარეობს, რომ გადეტერმინისტულების ყველა მეთოდში, რომელიც დღეისათვის არსებობს, მდგომარეობათა რიცხვი ექსპონენციურად იზრდება -- თუ არადეტერმინისტულ ავტომატს n სხვადასხვა მდგომარეობა აქვს, მის ექვივალენტურ დეტერმინისტულ ავტომატს 2^n მდგომარეობა *შეიძლება* ჰქონდეს, ანუ არადეტერმინისტული ავტომატი ზოგადად ექსპონენციური რიგით უფრო სწრაფი და პატარა შეიძლება იყოს.
იმის საჩვენებლად, თუ რამდენად შეიძლება გართულდეს არადეტერმინისტული ავტომატი დეტერმინიზაციის პროცესის შემდეგ, განვიხილოთ რაიმე ანბანი $A = \{a_1, a_2, ..., a_p\}$, სიტყვათა რაღაცა სიმრავლე $V = \{V_1, V_2, ..., V_n \mid V_i \in A^*\}$ და ენა $L = \{W \mid W$ სიტყვა V სიმრავლიდან ერთ სიტყვას მაინც შეიცავს$\}$. ამ ენის განმსაზღვრელი არადეტერმინისტული სასრული ავტომატი ქვედა ნახატშია მოყვანილი. აქ $V_1 = (w_{1,1}, ..., w_{1,k}), V_2 = (w_{2,1}, ..., w_{2,l}), ..., V_n = (w_{n,1}, ..., w_{n,m})$.



ნახ. 6: არადეტერმინისტული სასრული ავტომატის მაგალითი

ნახ. 7-ში ნაჩვენებია კონკრეტული მაგალითი ორი ავტომატისა -- არადეტერმინისტულის და დეტერმინისტულის, რომელიც ქართულ ანბანზე აგებული მონაცემისათვის გაგვცემს პასუხს შეკითხვაზე, შეიცავს თუ არა ეს მონაცემი სიტყვებს „თივა" ან „ათი"? აქ A ქართული ნბანია, ხოლო $/a$ ნიშნავს a სიმბოლოს გარდა ნებისმიერ ასოს.
აღსანიშნავია ის უმნიშვნელოვანესი გარემოება, რომ ჩვენ არადეტერმინისტული ავტომატის მუშაობის პროცესს ვერ აღვწერთ -- ვერ დავადგენთ, თუ როგორ ირჩევს არადეტერმინისტული სისტემა რამოდენიმე სწორი გზიდან საჭირო გზას. ჩვენთვის მთავარია, რომ ასეთ სისტემას ამის გაკეთება შეუძლია (ასეთი გზა არსებობს).
მნიშვნელოვანია ისიც, რომ არადეტერმინისტული ავტომატი *არასიმეტრიულია*, ანუ, თუ დეტერმინისტული ავტომატი რეკურსიულ გამოთვლად ამოცანებზე აუცილებლად ჩერდება რაღაცა პასუხით, არადეტერმინისტული ჩერდება დადებით პასუხზე, ხოლო უარყოფითზე შეიძლება არ გაჩერდეს -- თუ ჩვენს მაგალითში შემოვიდა სიტყვა „თითი", არადეტერმინისტული ავტომატი „ჩარჩება" მდგომარეობაში 0.

როგორც აღმოჩნდა, რეგულარულ ენათა კლასი ზუსტად ემთხვევა სასრულ ავტომატთა მიერ განსაზღვრულ ენათა კლასს: ნებისმიერი რეგულარული ენისათვის მოიძებნება სასრული ავტომატი, რომელიც ამ ენას განსაზღვრავს და პირიქით -- ნებისმიერი სასრული ავტომატის მიერ განსაზღვრული ენა რეგულარულია. ესე იგი,



ნახ. 7: ერთი და იგივე ენის განმსაზღვრელი არადეტერმინისტული და დეტერმინისტული ავტომატი

მოცემული ენა რეგულარულია მაშინ და მხოლოდ მაშინ, თუ მოიძებნება ისეთი სასრული ავტომატი, რომელიც ამ ენას განსაზღვრავს.

არსებობს უსასრულოდ ბევრი რეგულრული ენა, მაგრამ აგრეთვე უსასრულოდ ბევრი (არათვლადი სიმძლავრის) არარეგულარული ენა. ასეთი ენის მაგალითია $a^n b^n$, სადაც $n \in \mathbb{N}$. აქ პრობლემა ის არის, რომ სასრულმა ავტომატმა თავისი შეზღუდული რესურსით ნებისმიერად დიდი რაოდენობის ინფორმაცია უნდა დაიხსომოს -- n ხარისხი რაგინდ დიდი შეიძლება იყოს (ეს მკაცრად ე.წ. პამპინგ ლემის - Pumping Lemma - საშუალებით მტკიცდება). იგივე პრობლემა შეიძლება წამოიჭრას ჩვენს გარშემო არსებულ კომპიუტერებშიც -- საკმარისად დიდი n-თვის ნებისმიერი მეხსიერება გადაიტვირთება. ამ პრობლემის მოსაგვარებლად შემოიტანეს უსასრულოდ დიდი მეხსიერება -- სასრულ ავტომატს დაუმატეს ე.წ. LIFO (last-in-first-out) სტეკი (ბოლოს ჩაწერილი ინფორმაცია პირველ რიგში იკითხება) და მიიღეს სტეკიანი ავტომატი, რომელიც ე.წ. უკონტქსტო ენათა კლასს და მათ შორის ზემოთ აღნიშნულ ენასაც განსაზღვრავს.

ისეთი ენების მაგალითებად, რომლებიც *არაა უკონტექსტო*, შეიძლება მოვიყვანოთ:

$$\{a^p \mid p \text{ მარტივი რიცხვია}\};\ \ \{a^{n^2} \mid n \geq 0\};\ \ \{www \mid w \in \{a,b\}^*\};\ \ \{a^n b^n c^n \mid n \geq 0\}.$$

ეს მაგალითები იმაზე მიგვანიშნებს, რომ მხოლოდ უსასრულო მეხსიერების დამატება არაა საკმარისი ამოცანათა დიდი კლასის გადაჭრისათვის, რადგან შეუძლებელია ისეთი მარტივი ენის განსაზღვრა, როგორიცაა $\{a^n b^n c^n \mid n \geq 0\}$. ამ პრობლემის დასაძლევად შეიმუშავეს ე.წ. ტიურინგის მანქანები, რომლებიც, პრინციპში, სასრული ავტომატის მსგავსად მოქმედებენ, მაგრამ დამატებით უსასრულოდ დიდი მეხსიერება აქვთ, რომლიდან წაკითხვა ან ჩაწერა ყოველგვარი შეზღუდვის გარეშე შეუძლია (სტეკიანი ავტომატებისაგან განსხვავებით, რომლებსაც ერთ ჯერზე მხოლოდ ბოლოს ჩაწერილი ინფორმაციის წაკითხვა შეუძლიათ).
ერთი შეხედვით ტიურინგის მანქანები საკმაოდ პრომიტიულად შეიძლება მოგვეჩვენოს, მაგრამ, როგორც აღმოჩნდა, მათი გაძლიერების მცდელობას არანაირი საგრძნობი უპირატესობა არ მოაქვს -- გაძლიერებული მანქანებიც ზუსტად იმავე კლასის ენებს განსაზღვრავს, რაც ჩვეულებრივვი მანქანა, თანაც *თითქმის* იმავე სისწრაფით. როგორც ვარაუდობენ (ჩერჩის თეზისი), ტიურინგის მანქანებით გამოთვლილ ენათა კლასში ყველა გამოთვლადი ენა უნდა შედიოდეს, რაც იმას ნიშნავს, რომ ტიურინგის მანქანისა და ალგორითმის ცნება ერთი და იგივეა.

ძირითადად ტიურინგის მანქანები სამ კრიტერიუმს აკმაყოფილებენ:

(ა) ტიურინგის მანქანები უნდა აკმაყოფილებდნენ სასრული ავტომატების ძირითად პრინციპებს;

(ბ) ტიურინგის მანქანების ფორმალური აღწერა და პრინციპი უნდა იყოს მარტივი;

(გ) ტიურინგის მანქანა უნდა იყოს ზოგადი და აღწერდეს რაც შეიძლება მეტ გამოთვლად ენას.

ზოგადად, ტიურინგის მანქანა იგივე სასრული ავტომატია, რომელსაც დამატებული აქვს უსასრულოდ გრძელი მეხსიერების კვალი *(tape)* და ე.წ. წამკითხავ-ჩამწერი თავაკი. მეხსიერების კვალი დაყოფილია უჯრედებად. თითოეულ უჯრედში ჩაწერილია Σ ანბანის რაიმე ელემენტი ან ცარიელი სიმბოლო (ხარვეზი) $\sqcup$, ან მეხსიერების ყველაზე მარცხენა უჯრედში კვალის მარცხენა ბოლოს სიმბოლო $\triangleright$. ამრიგად, მეხსიერების კვალი მარცხნიდან შეზღუდულია, ხოლო მარჯვნივ კი უსასრულოდ გრძელდება. მანქანის შემავალი მონაცემი უშუალოდ $\triangleright$ სიმბოლოს მარჯვნივ წერია და მისი ბოლო ცარიელი სიმბოლოა (მეხსიერების ყველა დანარჩენი უჯრედიც ამ ცარიელი სიმბოლოთია შევსებული. მუშაობის დაწყების წინ წამკითხავი თავაკი მეხსიერების მარხენა კიდესთან, შემავალი

მონაცემის პირველ სიმბოლოზე დგას (თუ მონაცემები არ არის, ანუ მეხსიერების ყველა უჯრედი ცარიელია, მაშინ პირველ ცარიელ უჯრედზე).
სასრული ავტომატის მსგავსად, ტიურინგის მანქანაც თითო ბიჯში ერთი მდგომარეობიდან მეორეში გადადის. ამას გარდა, ჩამწერ-წამკითხავი თავაკი მეხსიერების ერთ-ერთ უჯრედზეა გაჩერებული. თუ რა მდგომარეობაში გადავა იგი თითოეულ ბიჯში, ეს იმაზეა დამოკიდებული, თუ რომელ მდგომარეობაშია მანქანა და რა სიმბოლოს კითხულობს ჩამწერ-წამკითხავი თავაკი (რა სიმბოლო წერია იმ უჯრედში, სადაც ეს თავაკია გაჩერებული). ერთი მდგომარეობიდან მეორეში გადასვლის გარდა, მანქანა Σ ანბანის რაღაცა გარკვეულ სიმბოლოს ჩაწერს მეხსიერების იმ უჯრაში, სადაც ჩამწერ-წამკითხავი თავაკია გაჩერებული და თავაკს ან დატოვებს იგივე უჯრედზე, ან გაწევს ან მარცხენა, ან მარჯვენა უჯრედზე.
როგორც აღვნიშნეთ, მეხსიერების კვალი მარცხნიდან შემოფარგლულია. იმისათვის, რომ ტიურინგის მანქანა მიხვდეს, სადაა მეხსიერების მარცხენა ბოლო, მას სპეციალური სიმბოლოთი აღნიშნავენ: $\triangleright$. როგორც კი ჩამწერ-წამკითხავი თავაკი ამ სიმბოლოს წაიკითხავს, მანქანა თავაკს ერთი უჯრედით მარჯვნივ გაწევს. მანქანაში შემავალი მონაცემები უშუალოდ მარცხენა ბოლოს სიმბოლოს შემდეგ იწერება, ხოლო დანარჩენი უჯრედები კი $\sqcup$ ცარიელი სიმბოლოთია შევსებული. გამოთვლებისას შესაძლებელია მონაცემების ადგილას სხვა სიმბოლოების ჩაწერა და უსასრულოდ ბევრი მეხსიერების გამოყენება.

მათემატიკურ ტერმინებში ტიურინგის მანქანა $TM = (K, \Sigma, \delta, s, H)$, სადაც

K არის ტიურინგის მანქანის მდგომარეობათა სასრული სიმრავლე;

Σ არის ანბანი, რომელიც შეიცავს ხარვეზს $\sqcup$ და ***მარცხენა ბოლოს*** სიმბოლოს $\triangleright$, მაგრამ არ შეიცავს სომბოლოებს $\leftarrow$, $\rightarrow$ და $\uparrow$ (თავაკი გაწიე მარცხნივ, გაწიე მარჯვნივ, დატოვე ადგილზე);

$s \in K$ არის საწყისი მდგომარეობა;

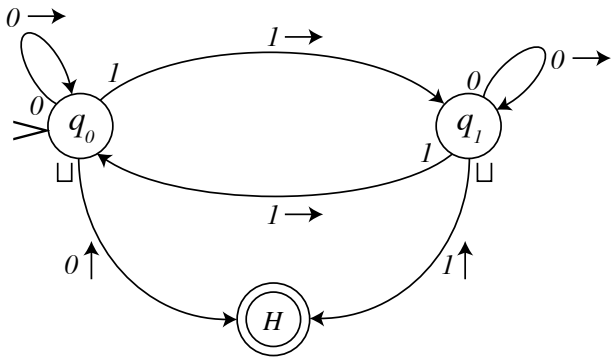
$H \subset K$ არის საბოლოო მდგომარეობათა სიმრავლე (თუ მანქანა ამ მდგომარეობაში გადავა, მუშაობა შეჩერდება);

$\delta : (K - H) \times \Sigma \to K \times \Sigma \times \{\leftarrow, \rightarrow, \uparrow\}$ არის მდგომარეობათა ფუნქცია, რომლისთვისაც

(ა) $\forall q \in K - H$, თუ $\delta(q, \triangleright) = (p, \triangleright, \rightarrow)$ (თუ მანქანა იმყოფება მდგომარეობაში q და მივაღწიეთ მეხსიერების მარცხენა ზღვარს, ერთი უჯრედით მარჯვნივ გადმოვდივართ);

(ბ) $\forall q \in K - H$ და $a \in \Sigma$, თუ $\delta(q, a) = (p, b, \{\leftarrow, \rightarrow, \uparrow\})$, მაშინ $b \neq \triangleright$ (თუ მანქანა იმყოფება მდგომარეობაში q და თავაკმა წაიკითხა Σ ანბანის რაღაცა სიმბოლო, შესაბამის უჯრედში ჩაიწერება რაღაცა სიმბოლო, რომელიც არ შეიძლება იყოს მეხსიერების მარცხენა ზღვარის სიმბოლო, შემდეგ თავაკი გადადის ან მარცხნივ, ან მარჯვნივ, ან რჩება ადგილზე და მანქანა გადადის რაღაცა მდგომარეობაში p).

ტიურინგის მანქანის ერთ-ერთი ყველაზე მარტივი მაგალითია „კენტობის შემოწმების" ავტომატი (parity checker), რომელიც ორობით კოდში ჩაწერილი რიცხვისათვის ბიტების ორობით ჯამს ითვლის: თუ ერთიანების რიცხვი კენტია, პასუხია 1, წინააღმდეგ შემთხვევაში კი -- 0 (ასეთი ალგორითმი სასრული ავტომატით ზემოთ განვიხილეთ).



ნახ. 8: „კენტობის შემოწმების" ტიურინგის მანქანა

მუშაობის დაწყებისას შემომავალი მონაცემი (ორობითი რიცხვი) მეხსიერერების კვალის უკიდურეს მარცხენა ნაწილში წერია (მის მარცხნივ მეხსიერების კვალის დასაწყისის მაჩვენებელი სიმბოლოა $\triangleright$). საწყისი მდგომარეობაა q_0. მანქანა ისეა მოწყობილი, რომ თუ მისი მდგომარეობაა q_0, ეს იმას ნიშნავს, რომ მან აქამდე ლუწი რაოდენობის 1 წაიკითხა. თუ მდგომარეობაა q_1, მაშინ მას აქამდე კენტი რაოდენობის 1 წაუკითხავს. ყოველი 1-ის წაკითხვისას მანქანა მდგომარეობას იცვლის.
ყოველ ბიჯზე მანქანა კითხულობს აქტუალურ უჯრედში ჩაწერილ მონაცემს, ამ მონაცემისა და აქტუალური მდგომარეობის მიხედვით წერს უჯრედში გარკვეულ სიმბოლოს, ამოძრავებს თავაკს გარკვეული მიმართულებით და გადადის შესაბამის მდგომარეობაში.

ჩვენს მაგალითში, იმ შემთხვევაში, თუ

- მანქანის მდგომარეობაა q_0:
  - თუ მანქანამ წაიკითხა სიმბოლო 0 (წაკითხული სიმბოლო წერია შესაბამისი გამომავალი ისრის დასაწყისში), შესაბამის უჯრაში იწერება სიმბოლო 0 (ანუ, ფაქტიურად, იგივე რჩება -- ჩაწერილი სიმბოლო წერია გარდამავალი ისრის შუაში), მანქანაც რჩება მდგომარეობაში q_0 (რომელიც წრეშია ჩაწერილი) და თავაკი მოძრაობს მარჯვნივ (ემზადება შემდეგი სიმბოლოს წასაკითხად -- თავაკის მოძრაობის მიმართულება აღნიშნულია ჩასაწერი სიმბოლოს გვერდით).
  - თუ წაკითხული სიმბოლოა 1, მაშინ მანქანა წერს სიმბოლოს 1, თავაკს ამოძრავებს მარჯვნივ და გადადის მდგომარეობაში q_1. თუ წაკითხული სიმბოლოა $\sqcup$ (რაც შემომავალი რიცხვის ბოლოს ნიშნავს), მაშინ იწერება სიმბოლო 0, თავაკი მოძრაობს მარჯვნივ, მანქანა გადადის საბოლოო მდგომარეობაში H და ჩერდება.
- მანქანის მდგომარეობაა q_1:
  - თუ მანქანამ წაიკითხა სიმბოლო 0, შესაბამის უჯრაში იწერება სიმბოლო 0, მანქანაც რჩება მდგომარეობაში q_1 და თავაკი მოძრაობს მარჯვნივ.
  - თუ წაკითხული სიმბოლოა 1, მაშინ მანქანა წერს სიმბოლოს 1, თავაკს ამოძრავებს მარჯვნივ და გადადის მდგომარეობაში q_0. თუ წაკითხული სიმბოლოა $\sqcup$, მაშინ იწერება სიმბოლო 1, თავაკი მოძრაობს მარჯვნივ, მანქანა გადადის საბოლოო მდგომარეობაში H და ჩერდება.

მანქანის შეჩერების შემდეგ საბოლოო პასუხი იმ უჯრედში წერია, სადაც გაჩერებულია თავაკი.

ფორმალურად ზემოთ მოყვანილი მანქანა შემდეგნაირად ჩაიწერება:

$K = \{q_0, q_1, H\}$ --- მდგომარეობათა სიმრავლე;

$\Sigma = \{\sqcup, \triangleright, 0, 1\}$ --- ანბანი;

$s = q_0$ --- საწყისი მდგომარეობა;

H --- საბოლოო მდგომარეობა;

$\delta : (q_0, 0) \mapsto (q_0, 0, \rightarrow)$

$\delta : (q_0, 1) \mapsto (q_1, 1, \rightarrow)$

$\delta : (q_0, \sqcup) \mapsto (H, 0, \uparrow)$

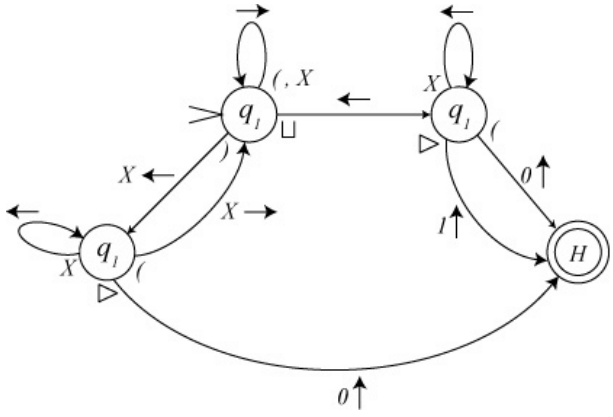
$\delta : (q_1, 0) \mapsto (q_1, 0, \rightarrow)$

$\delta : (q_1, 1) \mapsto (q_0, 1, \rightarrow)$

$\delta : (q_1, \sqcup) \mapsto (H, 1, \uparrow)$

შედარებით უფრო რთული მანქანის მაგალითია ე.წ. ბრჩხილების დასმის შემოწმების ავტომატი, რომელიც ბრჩხილების დასმის სინტაქსს ამოწმებს. მაგალითად, „(()()" არასწორადაა დასმული, ხოლო „(()(()())(()())" კი -- სწორად.
სიტყვიერად ამის ალგორითმი შემდეგნაირად აღიწერება: იმოძრავე მარჯვნივ პირველ დახურულ ბრჩხილამდე „)" ან ცარიელ სიმბოლომდე $\sqcup$, რომელიც შემომავალი მონაცემის ბოლოს აღნიშნავს. შეცვალე პირველივე

დახურული ბრჩხილი სიმბოლოთი X. ამის შემდეგ იმოძრავე მარცხნივ პირველ გახსნილ ბრჩხილამდე „(" ან მეხსიერების კვალის მარცხენა კიდემდე $\triangleright$. ჯერ მარჯვენა და მერე მარცხენა ბრჩხილის X სიმბოლოთი შეცვლა ბრჩხილების შესაბამისი წყვილის მოშლას ნიშნავს და დარჩენილ ბრჩხილებს შორის უნდა შემოწმდეს სისწორე. თუ მარცხნივ მოძრაობისას გახსნილი ბრჩხილი არ შეგვხვდა, შეცდომაა (იმიტომ, რომ დახურული ბრჩხილი არსებობდა, მის მარცხნივ გახსნილი კი -- არა). თუ შესაბამისად მარჯვნივ მოძრაობისას დახურული ბრჩხილი აღარ შეგვხვდა (მივაღწიეთ მონაცემის ბოლოს -- ცარიელ სიმბოლოს), მაშინ ეს იმას ნიშნავს, რომ ყველა დახურული ბრჩხილი შეიცვალა X სიმბოლოთი და მხოლოდ იმის გადამოწმებაა საჭირო, გვხვდება თუ არა დარჩენილ სიმბოლოთა შორის X სიმბოლოს გარდა გახსნილი ბრჩხილი? თუ გვხვდება, მაშინ დაშვებული იყო შეცდომა, მანქანა ბეჭდავს 0 და ჩერდება. თუ არა, მაშინ შეცდომა დაშვებული არ იყო, მანქანა ბეჭდავს 1 და ჩერდება. ორივე შემთხვევაში პასუხი იმ ადგილას იქნება დაწერილი, სადაც თავაკი საბოლოოდ გაჩერდება. ნახ. 9-ში მოყვანილია „დაბრჩხილვის შემოწმების" ტიურინგის მანქანა.



ნახ. 9: „დაბრჩხილვის შემოწმების" ტიურინგის მანქანა

რადგან ტიურინგის მანქანას აქვს უსასრულოდ დიდი მეხსიერების კვალი, მას შეუძლია უსასრულოდ დიდი სიტყვის ჩაწერა და, აქედან გამომდინარე, მეტის გაკეთება, ვიდრე მხოლოდ „კი" ან „არა" პასუხის გაცემა სხვადასხვა შეკითხვაზე.

თუ მოცემულია რაიმე ანბანი Σ და ფუნქცია $f : \Sigma^* \to \Sigma^*$, მაშინ *შეიძლება* არსებობდეს ისეთი ტიურინგის მანქანა M, რომელიც ამ ფუნქციას „გამოითვლის", ანუ $\forall w \in \Sigma^*$, w მონაცემით გაშვებული M ტიურინგის მანქანა გამოთვლას სასრულ დროში დაამთავრებს და მისი მეხსიერების კვალზე ეწერება $\triangleright f(w)\sqcup$. ისეთ ფუნქციებს, რომელთათვისაც ამდაგვარი ტიურინგის მანქანები არსებობს, *რეკურსიული* ეწოდება.

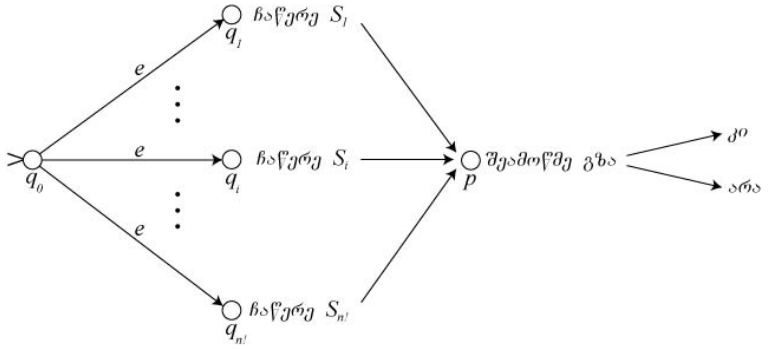
თუ მოცემულია რაღაცა Σ ანბანზე არსებული ენა L და რაღაცა სიტყვა $w \in \Sigma^*$, შესაძლებელია არსებობდეს ისეთი ტიურინგის მანქანა T, რომელიც $\forall w \in L$, w მონაცემზე ჩერდება და იძლევა დადებით პასუხს, ხოლო თუ $w \notin L$ შეიძლება არც შეჩერდეს და უსასრულოდ გააგრძელოს გამოთვლა (ამ შემთხვევაში იტყვიან, რომ T ნაწილობრივ განსაზღვრავს L ენას). ასეთ ენებს რეკურსიულად გადათვლადი ეწოდება. თუ T შეჩერდება აგრეთვე იმ შემთხვევაში, თუ $w \notin L$ და მოგვცემს უარყოფით პასუხს, ასეთ ენას რეკურსიულს უწოდებენ. ცხადია, რომ რეკურსიულად გადათვლადი ენების სიმრავლე რეკურსიულ ენათა სიმრავლეს მოიცავს. საინტერესოა შემდეგი შეკითხვა: შეიძლება თუ არა ისეთი მანქანის გარდაქმნა, რომელიც რაღაცა ენას ნაწილობრივ განსაზღვრავს, ისე, რომ მივიღოთ ისეთი ტიურინგის მანქანა, რომელიც იგივე ენას განსაზღვრავს? სხვა სიტყვებით რომ ვთქვათ, იარსებებს თუ არა ისეთი ალგორითმი, რომელიც ყველა რეკურსიულად გადათვლადი ენისათვის ყოველთვის შეჩერდება? როგორც აღმოჩნდა, არსებობს ისეთი რეკურსიულად გადათვლადი ენები, რომელიც არაა რეკურსიული, ანუ ასეთი ალგორითმის შედგენა შეუძლებელია. ეს არის ალგორითმების თეორიის ერთ-ერთი ცენტრალური შედეგი.

როგორც სასრულ ავტომატებში, ასევე ტიურინგის მანქანებში, ყოველი ამოცანისათვის ცალკე უნდა შეიქმნას მანქანა ან ავტომატი (ისევე, როგორც ცალკეული ამოცანისათვის ცალკეული პროგრამა და ალგორითმი უნდა შეიქმნას). მაგრამ არსებობს აგრეთვე ე.წ. ***უნივერსალური*** ტიურინგის მანქანა, რომესაც ნებისმიერი სხვა ტიურინგის მანქანის სიმულაცია შეუძლია: იგი მონაცემებთან ერთად იმ მანქანის აღწერილობას იღებს, რომლის სიმულაციაცაა საჭირო და შემდგომში ნებისმიერი მონაცემისათვის ამ მანქანასავით მოიქცევა -- ამის ანალოგია კომპილატორები და ჩვენს გარშემო არსებული კომპიუტერები, რომლებიც პროგრამების საშუალებით სხვადასხვა ფუნქციებს ითვლიან.

შემდგომლში ჩვენ ასეთ უნივერსალურ მანქანებს განვიხილავთ.

როგორც აღმოჩნდა, გამომთვლელის ზემოთ მოყვანილი „პრიმიტიული" მოდელის ბევრად გაუმჯობესება არ შეიძლება: მეხსიერების პარალელური კვალების დამატება, მეხსიერების ნებისმიერ უჯრაზე ერთ ბიჯში გადასვლის შესაძლებლობა (ჩვენს გარშემო არსებულ კომპიუტერთა RAM მოდელის შემოღება), ორმხრივად შეუზღუდავი მეხსიერება, ორგანზომილებიანი მეხსიერების დამატება და სხვა მეთოდები რაიმე საგრძნობ უპირატესობას არ გვმატებს: გამოთვლად ფუნქციათა კლასი იგივე რჩება და სისწრაფე მხოლოდ პოლინომიური ფაქტორით იზრდება, რაც იმას ნიშნავს, რომ ექსპონენციური ზრდის რიგის მქონე ალგორითმები ისევე ექსპონენციური რჩება, ხოლო პოლინომიური -- ისევე პოლინომიური.

მაგრამ თვისობრივად განსხვავებულ უპირატესობას n სიგრძის მონაცემზე გამოთვლისათვის საჭირო ბიჯების თვალსაზრისით ***არადეტერმინისტულობა*** მატებს -- ისეთი, როგორიც ჩვენ სასრულ ავტომატებში განვიხილეთ. ამის სადემონსტრაციოდ განვიხილოთ მოგზაური ბიზნესმენის ამოცანა (Traveling Salesman Problem, მოკლედ TSP), რომელშიც მოცემულია შეწონილი გრაფი G და რაღაცა ნატურალური რიცხვი K. არსებობს თუ არა ამ გრაფში ისეთი გზა, რომელიც გაივლის ყველა კვანძზე და გავლილი წიბოების წონათა ჯამი $S \leq K$? (თუ განვიხილავთ გრაფს, როგორც ქალაქებს და მათ შორის გზებს შესაბამისი მანძილებით, დაადგინეთ, შეუძლია თუ არა ბიზნესმენს, შემოიაროს ყველა ქალაქი ისე, რომ K კილომეტრზე ნაკლები მანძილი გაიაროს?)
ეს კლასიკური ამოცანა (როგორც მრავალი სხვა) იმის მაგალითია, თუ როგორი ეფექტური შეიძლება იყოს არადეტერმინისტული მანქანა დეტერმინისტულთან შედარებით.
თუ ტიურინგის მანქანა მონაცემად იღებს გრაფის აღწერილობას GW (კვანძების სიმრავლე $V = \{v_1, ..., v_n\}$ და წიბოების სიმრავლე E თავისი შესაბამისი წონით), მისი სიგრძე იქნება $|GW| = O(n^2)$. განვიხილოთ $\{v_1, ..., v_n\}$ რიცხვების პერმუტაციების (გადანაცვლებების) სიმრავლე $S = \{S_1, ..., S_{n!}\}$ (თითოეული ასეთი პერმუტაციის ჩანაწერის სიგრძე არ აღემატება n^2, ანუ თვით მონაცემის სიგრძეს $|GW|$.
არადეტერმინისტული მანქანა წაიკითხავს მონაცემს და დაადგენს გრაფში წვეროების რაოდენობას. n წვეროსათვის იგი ნახ. 10-ში მოყვანილი ტიურინგის მანქანის სიმულაციას დაიწყებს:



ნახ. 10: TSP ამოცანის არადეტერმინისტული ტიურინგის მანქანის დიაგრამა

დასაწყისში მანქანის მდგომარეობაა q_0 და იგი ცარიელი სიტყვით *არადეტერმინისტულად* გადახტება ერთ-ერთ მდგომარეობაში q_i, რომელშიც მეხსიერებაში ჩაწერს S_i პერმუტაციას. შემდეგ გადავა მდგომარეობაში p და შეამოწმებს, არის თუ არა ეს პერმუტაცია რამე შეკრული გზა გრაფში (შემოუვლის თუ არა ყველა წვეროს). თუ არა, გადავა საბოლოო მდგომარეობაში „არა". თუ კი, შეამოწმებს, არის თუ არა ამ გზის სიგრძე ნაკლები ან ტოკი K. თუ კი, გადავა მდგომარეობაში „კი", წინააგმდეგ შემთხვევაში -- მდგომარეობაში „არა". ადვილი სანახავია, რომ ყველა შესაძლო გზისათვის, რომლის ზომაცაა ნაკლები ან ტოლი K, ეს მანქანა გადავა მდგომარეობაში „კი". იმის გამო, რომ ოპერაციები „ჩაწერე S_i" და „შეამოწმე გზა" ორკვალიან ტიურინგის მანქანაზე $O(|GW|^2)$ დროში ხერხდება, მთელი გამოთვლის პროცესიც ამ დროში ჩაეტევა.
აღსანიშნავია, რომ, სასრული ავტომატების მსგავსად, ზემოთ ნახსენები არადეტერმინისტული ასიმეტრიულობა ტიურინგის მანქანებშიც იჩენს თავს.

1.6 არაამოხსნადი ამოცანები

როგორც ზემოთ აღვნიშნეთ, არსებობს რეკურსიულად გადათვლადი ენები, ანუ ისეთი ენები, რომლის განმსაზღვრელი ტიურინგის მანქანა შეჩერდება მაშინ, თუ მისი მონაცემი ამ ენის სიტყვაა და შეიძლება საერთოდ არ შეჩერდეს, თუ მისი მონაცემი ამ ენის სიტყვა არაა. აქედან გამომდინარე, შეიძლება მოხდეს ისეთი რამ, რომ მოცემული ტიურინგის მანქანა TM რაღაცა მონაცემებზე შეჩერდეს, რაღაცაზე კი -- არა.

ბუნებრივია შემდეგი შეკითხვა: შესაძლებელია თუ არა ისეთი ალგორითმის შექმნა, რომელიც მონაცემად მიიღებს რაიმე T ტიურინგის მანქანის აღწერილობას, რაიმე მონაცემს X და **სასრულ დროში** განსაზღვრავს, შეჩერდება თუ არა T მანქანა X მონაცემზე.
ზოგ შემთხვევაში ამის გაკეთება დიდ სიძნელეს არ წარმოადგენს -- გარკვეული ანალიზით შეიძლება იმის დადგენა, ჩავარდება თუ არა კონკრეტული პროგრამა უსასრულო ციკლში, მაგრამ პრობლემა ისაა, რომ უსასრულოდ ბევრ ამოცანას თავისებური მიდგომა ჭირდება.

იმის დასამტკიცებლად, რომ ასეთი *ზოგადი* ალგორითმი არ არსებობს, დავუშვათ საწინააღმდეგო: მოცემულია ტიურინგის მანქანა H, რომელიც ნებისმიერი სხვა ტიურინგის მანქანისათვის TM და მონაცემისათვის X იძლევა პასუხს შეკითხვაზე, შეჩერდება თუ არა TM მანქანა X მონაცემზე: ჩვენ მას ვაძლევთ TM მანქანის აღწერილობას და მონაცემს X: „TM, X", H(„TM, X") გვაძლევს პასუხს „კი" ან „არა".

ქვემოთ მოყვანილია დამტკიცების ძირითადი იდეა:

დავუშვათ, რომ არსებობს მეორე ტიურინგის მანქანა N, რომელიც იგივე მონაცემზე შეჩერდება მაშინ და მხოლოდ მაშინ, თუ H ამ მონაცემზე *გვაძლევს პასუხს „არა"*:

- თუ TM შეჩერდება მონაცემზე X, ანუ $H(TM, X) =$„კი" $\Rightarrow N(X)$ არ შეჩერდება;
- თუ TM არ შეჩერდება მონაცემზე X, ანუ $H(TM, X) =$„არა" $\Rightarrow N(X)$ შეჩერდება.

ცხადია, რომ ამ N მანქანასაც თავისი აღწერილობა უნდა ჰქონდეს: „W_N".

ახლა დავსვათ შემდეგი ამოცანა: შეჩერდება თუ არა $N(W_N)$?

- თუ $N(W_N)$ შეჩერდება, ეს იმას ნიშნავს, რომ $H(N, W_N)$ არ შეჩერდება და, H მანქანის განსაზღვრების თანახმად, $N(W_N)$ არ შეჩერდება.
- თუ $N(W_N)$ არ შეჩერდება, ეს იმას ნიშნავს, რომ $H(N, W_N)$ შეჩერდება და, H მანქანის განსაზღვრების თანახმად, $N(W_N)$ შეჩერდება.

  ორივე შემთხვევაში ვაწყდებით წინააღმდეგობას.

არაამოხსნად ამოცანათა რამოდენიმე მაგალითად შეგვიძლია მოვიყვანოთ:

ინფორმატიკიდან:

- ნებისმიერი ტიურინგის მანქანისათვის განსაზღვრეთ, შეჩერდება თუ არა იგი ცარიელი მეხსიერებით გაშვებისას (თუ შემავალი მონაცემი არ არსებობს)?
- მოცემულია ორი ტიურინგის მანქანა M_1 და M_2. $L(M_1) = L(M_2)$?
- ნებისმიერი ტიურინგის მანქანისათვის განსაზღვრეთ, შეჩერდება თუ არა იგი ყველა მონაცემზე?
- ნებისმიერი ტიურინგის მანქანისათვის განსაზღვრეთ, შეჩერდება თუ არა იგი რაიმე მონაცემზე?
- მოცემულია ტიურინგის მანქანა M. როგორია მის მიერ ნაწილობრივ განსაზღვრული ენა -- რეგულარული, უკონტქსტო, თუ რეკურსიული?
- არის თუ არა მოცემული პროგრამა ყველაზე მოკლე იმ პროგრამათა შორის, რომელიც ერთსა და იმავე მონაცემზე ერთნაირ პასუხს გვაძლევს?

მათემატიკიდან:

- სიტყვის ამოცანა ჯგუფებსა და ნახევარჯგუფებში;
- სასრულად აღწერილი 4 განზომილებიანი მრავალნაირობების ექვივალენტურობა;
- აღწერს თუ არა მიმართებათა ორი სიმრავლე ერთსა და იმავე ჯგუფს ან ნახევარჯგუფს?
- აქვს თუ არა რეალური ფესვი ისეთ გამოსახულებას, რომელიც ტრიგონომეტრიულ და ალგებრულ ფუნქციებს შეიცავს?

მნიშვნელოვანია შემდეგი შედეგი: თუ Σ ანბანზე შექმნილი L_1 ენა არაამოხსნადია (ანუ არ არსებობს ტიურინგის მანქანა, რომელიც მას განსაზღვრავს) და იგი რეკურსიულად დაიყვანება ამავე ანბანზე შექმნილ L_2 ენაზე (ანუ არსებობს ისეთი რეკურსიული ფუნქცია $\tau : \Sigma^* \to \Sigma^*$, რომ $w \in L_1 \Rightarrow \tau(w) \in L_2$), მაშინ L_2 არაა ამოხსნადი.

სავარჯიშო 1.1: დაამტკიცეთ ზემოთ ჩამოყალიბებული დებულება

ბევრი ამოცანის არაამოხსნადობა ე.წ. რაისის თეორემიდან გამომდინარეობს (Rice theorem), რომლის მიხედვითაც ტურინგის მანქანების შესახებ მნიშვნელოვანს ვერაფერს დავამტკიცებთ. როგორც ჩანს, რაისის თეორემა ფუნდამენტურია იმ თვალსაზრისით, რომ ადამიანის მიერ „მნიშვნელოვანი" საკითხების გადაჭრის შეუძლებლობას ეხება.

1.7 რაისის თეორემა არაამოხსნადობის შესახებ

რაისის თეორემა (ზოგიერთ წყაროში რაის-მიჰილ-შაპიროს თეორემა - Rice-Myhill-Shapiro theorem) გვეუბნება, რომ არ არსებობს ზოგადი მეთოდი, რომლითაც *ნებისმიერი* ალგორითმისთვის (ტიურინგის მანქანისთვის) დავადგენთ, ითვლის თუ არა იგი რაიმე კონკრეტულ, წინასწარ განსაზღვრულ შედეგს (ანუ გამოითვლის თუ არა მოცემული პროგრამა მოცემულ ფუნქციას) გარდა ორი ტრივიალური მაგალითისა. ეს კი იმას ნიშნავს, რომ ყოველ ალგორითმს (ან ალგორითმთა კლასს) ცალკე მეთოდი უნდა მოვუძებნოთ, თუ როგორ გავაანალიზოთ მისი ფუნქცია.
გარდა იმისა, რომ ამ თეორემის შედეგად ძალიან ბევრი არაამოხსნადი ამოცანა აღმოაჩინეს, მისი შედეგები უფრო ღრამა და ერთობ ფილოსოფიურია: ადამიანის აზროვნება მნიშვნელოვან (გლობალურ) საკითხებს ვერ ჩაწვდება.

1.7.1 განსაზღვრება და ძირითადი შედეგები

განმარტება 1.1: მოცემული TM ტიურინგის მანქანის აღწერილობა ორობით ანბანზე აღვნიშნოთ როგორც $\langle TM \rangle$.
$L_{TM} = \{\langle M \rangle \,|M$ ტიურინგის მანქანაა$\}$ ყველა არსებული ტიურინგის მანქანის აღწერილობაა.
თუ $\mathcal{C}$ ენათა რაღაც სიმრავლეა, $L_{\mathcal{C}} = \{\langle M \rangle \,|L(M) \in \mathcal{C}\}$ ისეთ ტიურინგის მანქანათა აღწერილობების სიმრავლეა, რომლებიც $\mathcal{C}$ სიმრავლის ენებს განსაზღვრავენ.
ასეთ $L_{\mathcal{C}}$ სიმრავლეს (ენას) ეწოდება ტრივიალური ტიურინგის მანქანათა შესახებ, თუ $L_{\mathcal{C}} = \emptyset$ ან $L_{\mathcal{C}} = L_{TM}$. წინააღმდეგ შემთხვევაში მას ტიურინგის მანქანათა შესახებ არატრივიალური ენა ეწოდება.

სხვა სიტყვებით რომ ვთქვათ, ენა არატრივიალურია, თუ არსებობს ორი ისეთი ტიურინგის მანქანა, რომელთაგან ერთი ამ სიმრავლეს ეკუთვნის და მეორე კი - არა.

სავარჯიშო 1.2: განიხილეთ ზემოთ მოყვანილი განსაზღვრება $L_{\mathcal{C}} = \{\langle M \rangle \,|L(M) \in \mathcal{C}\}$. რისი ტოლი უნდა იყოს $\mathcal{C}$, რომ $L_{\mathcal{C}} = L_{TM}$?

მაგალითი 1.1: არატრივიალური ენებია
$L_{Reg} = \{\langle M \rangle \,|L(M)$ რეგულარულია$\}$
$L_{\emptyset} = \{\langle M \rangle \,|L(M) = \{\emptyset\}$ (ცარიელი ენა)$\}$
$L_{\Sigma} = \{\langle M \rangle \,|L(M) = \{\Sigma^*\}$ (მოცემულ Σ ანბანზე შედგენილ ყველა სიტყვათა სიმრავლე)$\}$
$L_{\epsilon} = \{\langle M \rangle \,|\, \epsilon \in L(M)\}$
$L_{CF} = \{\langle M \rangle \,|L(M)$ უკონტექსტოა$\}$

სავარჯიშო 1.3: დაამტკიცეთ ზემოთ მოყვანილ ენათა არატრივიალურობა.

რაისის თეორემა 1.1: ნებისმიერი $\mathcal{S}$ არატრივიალური ენა ტიურინგის მანქანების შესახებ არ არის გამოთვლადი.

რაისის თეორემით მტკიცდება, რომ თუ მოცემულია რაიმე სიტყვა, მასზე მხოლოდ იმის დადგენა შეგვიძლია, არის თუ არა იგი ტიურინგის მანქანის აღწერა. იმის დადგენა, თუ რას გამოითვლის მოცემული ტიურინგის მანქანა ან გამოითვლის თუ არა მოცემული ალგორითმი რაიმე კონკრეტულ ფუნქციას, შეუძლებელია.

დამტკიცება: ძირითადი იდეა სიტყვის ამოცანის ამ პრობლემაზე დაყვანაა: თუ მოცემულია $\langle M, w \rangle$ რაიმე ტიურინგის მანქანის და მისი მონაცემის აღწერა, ავაგებთ ისეთ ტიურინგის მანქანას M', რომ $w \in L(M) \Leftrightarrow L(M') \in \mathcal{S}$.

რადგან $\mathcal{S}$ არატრივიალურია, უნდა არსებობდეს ორი ენა $L_1 \in \mathcal{S}$ და $L_2 \notin \mathcal{S}$. ზოგადობის შეუზღუდავად შეგვიძლია დავასკვნათ, რომ $L_2 = \emptyset$, რადგან თუ $\emptyset \in \mathcal{S}$, არატრივიალურ ენად შეგვიძლია გამოვაცხადოთ $\overline{\mathcal{S}}$.
განვიხილოთ ტიურინგის მანქანა M_L ისეთი, რომ $L(M_L) = L_1$. მისი და $\langle M, w\rangle$ აღწერილობის გამოყენებით შევქმნათ ახალი ტიურინგის მანქანა M', რომელიც შემდეგ ოპერაციებს ატარებს:

M' მონაცემით x

- მოახდინე M მანქანის სიმულაცია w მონაცემზე;
- თუ M მიიღებს w სიტყვას, მოახდინე M_L მანქანის სიმულაცია x მონაცემზე;
- თუ M_L მიიღებს x სიტყვას, გადადი მდგომარეობაში "YES" (მიიღე w სიტყვა)

შენიშნვა: მოცემულ აღწერილობაში მნიშვნელოვანია, რომ თუ M მანქანა სასრულ დროში მიიღებს w სიტყვას და ამავდროულად M_L მანქანაც სასრულ დროში მიიღებს x სიტყვას, მაშინ M' მანქანა მიიღებს სიტყვას x. წინააღმდეგ შემთხვევაში მისი გამოთვლის პროცესი უსასრულოდ გაგრძელდება.

განვიხილოთ შემთხვევა, როდესაც M მანქანა w სიტყვას მიიღებს. მაშინ M' მიიღებს სიტყვას x მაშინ და მხოლოდ მაშინ, თუ M_L მიღებს სიტყვას x. აქედან გამომდინარე, $L(M') = L_1 \in \mathcal{S}$.
მაგრამ თუ M მანქანა w სიტყვას არ მიიღებს, მაშინ M' ვერც ერთ სიტყვას ვერ მიიღებს, რადგან M_L მანქანის სიმულაცია არ მოხდება და, აქედან გამომდინარე, $L(M') = \emptyset \notin \mathcal{S}$.

ზემოთ თქმულიდან ადვილად შეიძლება დავინახოთ, რომ თუ M' მანქანა $\mathcal{S}$ სიმრავლის ენას განსაზღვრავს, მაშინ M მანქანა შეჩერებულა x მონაცემზე და თუ M' $\overline{\mathcal{S}}$ სიმრავლის ენას (ამ შემთხვევაში ცარიელ ენას) განსაზღვრავს, M მანქანა x მონაცემზე არ შეჩერებულა. აქედან გამომდინარე, ტიურინგის მანქანებზე არატრივიალური ენის განსაზღვრა რომ ყოფილიყო შესაძლებელი, ზემოთ აღწერილი მეთოდებით M მანქანიდან M' მანქანას შევქმნიდით და $L(M') \in S$ საკითხის ამოხსნით M მანქანის შეჩერების საკითხიც ამოხსნადი იქნებოდა, რითაც წინააღმდეგობას ვიღებთ.

რ.დ.გ.

რაისის თეორემის შედეგები: შემდეგი ამოცანები არაამოხსნადია:

- მოცემულია ორი ტიურინგის მანქანა M_1 და M_2. $L(M_1) = L(M_2)$? შენიშვნა: ეს ე.წ. რაისის გაფართოებული თეორემის შედეგია, რომელიც შემდგომში იქნება ჩამოყალიბებული
- მოცემულია ორი ფორმალური გრამატიკა G_1 და G_2. $L(G_1) = L(G_2)$? შენიშვნა: იგივე, რაც ზემოთ
- მოცემული ტიურინგის მანქანისთვის გაარკვიეთ, თუ რომელ ენას გამოითვლის იგი
- ექნება მოცემული ტიურინგის მანქანის მიერ გამოთვლილ ფუნქციას რაიმე მონაცემზე მნიშვნელობა 1?
- გამოითვლის თუ არა მოცემული ტიურინგის მანქანა ერთსა და იმავე შედეგს ყოველ მონაცემზე?
- მიიღებს თუ არა მოცემული ტიურინგის მანქანა ყველა მონაცემს?
- მოცემული L ენისთვის განსაზღვრეთ, არის თუ არა იგი რეგულარული
- მოცემული L ენისთვის განსაზღვრეთ, არის თუ არა იგი უკონტექსტო
- მოცემული L ენისთვის განსაზღვრეთ, განსაზღვრავს თუ არა მას რომელიმე ტიურინგის მანქანა

მაგალითი 1.2: განვიხილოთ შემდეგი ამოცანა: მოცემულია ტიურინგის მანქანა M. არის თუ არა $L(M)$ პალინდრომების ენა ($w \in L(M) \Leftrightarrow w = w^R$)?
რაისის თეორემის გამოსაყენებლად ეს საკითხი შესაბამის ტერმინებში უნდა ჩამოვაყალიბოთ. პირველ რიგში ჩამოვაყალიბოთ პალინდრომული ენა

$$Pal = \{\langle M\rangle \,|L(M) \text{ შეიცავს ყველა პალინდრომს}\}$$

ცხადია, რომ ეს ენა არაა ტრივიალური და რაისის თეორემის თანახმად არაამოხსნადი უნდა იყოს.

რ.დ.გ.

სავარჯიშო 1.4: დაამტკიცეთ, რომ ზემოთ მოყვანილი ენა არაა ტრივიალური (მინიშნება: მოიყვანეთ ორი ტიურინგის მანქანის მაგალითი. ერთი იღებს მხოლოდ პალინდრომებს და მეორე ისეთ სიტყვასაც, რომელიც პალინდრომი არ არის).

სავარჯიშო 1.5: დაამტკიცეთ ზემოთ მოყვანილი ამოცანების არაამოხსნადობა.

მაგალითი 1.3: რაისის თეორემის მეშვეობით იმის ჩვენებაც შეიძლება, რომ ამოცანათა სირთულის ავტომატური შტკიცება შეუძლებელია. განვიხილოთ ენათა შემდეგი კლასი:

$$P = \{L | L \text{ ენა ამოიხსნება დეტერმინისტული ტიურინგის მანქანის მიერ პოლინომურ დროში}\}$$

სავარჯიშო 1.6: ჩაწერეთ ამ ამოცანის პირობა რაისის თეორემის ტერმინებში და დაამტკიცეთ P კლასის არაამოხსნადობა.

აქედან გამომდინარე, არაამოხსნადია შემდეგი ამოცანა:

მოცემულია ტიურინგის მანქანა M. შეიძლება თუ არა $L(M)$ ენის პოლინომურ დროში განსაზღვრა?

1.7.2 გასათვალისწინებელი პრობლემები

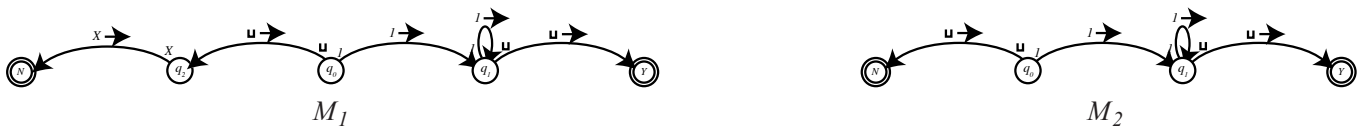
შესაძლებელია ისეთი ამოცანების წამოჭრა, რომლებსაც რაისის თეორემა არ ესადაგება, თუმცა ამის დანახვა მარტივი არაა. ასეთ შემთხვევებში მიღებული მტკიცება ან მთლიანად არასწორია, ან გარკვეულ უზუსტობებს შეიცავს. ამდაგვარი პრობლემების თავიდან ასაცილებლად განვიხილოთ რამოდენიმე კონკრეტული მაგალითი.

მაგალითი 1.4: ტიურინგის მანქანის მდგომარეობას, რომელიც არაა სამიზნე, ვუწოდოთ "ჩვეულებრივი". განვიხილოთ შემდეგი გადაწყვეტილების ამოცანა:
მოცემული M ტიურინგის მანქანისთვის დაადგინეთ, არსებობს თუ არა ისეთი მონაცემი w, რომლითაც M მანქანა გამოთვლის პროცესში ყველა ჩვეულებრივ მდგომარეობას ერთხელ მაინც გაივლის.

შეიძლება ვიფიქროთ, რომ თვისება „რომელიმე w მონაცემისთვის M მანქანა გამოთვლის პროცესში ყველა ჩვეულებრივ მდგომარეობას ერთხელ მაინც გაივლის" არაა ტრივიალური და რაისის თეორემის თანახმად ეს ამოცანა არ უნდა იყოს გამოთვლადი.
იმის მიუხედავად, რომ ეს ამოცანა მართლაც არაა ამოხსნადი, ეს ფაქტი რაისის თეორემის პირობებში ვერ ჯდება: თვით თვისების კონცეფცია აქ არასწორადაა გამოყენებული.
ეს გადაწყვეტილების ამოცანა შეიძლება შემდეგი ენის სახით ჩამოვაყალიბოთ:

$$T_{TM} = \left\{ \langle M \rangle \;\middle|\; \begin{array}{l} \text{რომელიმე } w \text{ მონაცემისთვის } M \text{ მანქანა} \\ \text{გამოთვლის პროცესში ყველა ჩვეულებრივ} \\ \text{მდგომარეობას ერთხელ მაინც გაივლის} \end{array} \right\}$$

ზემოთ ნახსენები გამონათქვამი არ აღწერს ენათა კლასს ტიურინგის მანქანათა შესახებ, რაც რაისის თეორემის პირობის აუცილებელი ნაწილია. ეს უფრო ტიურინგის მანქანის კონკრეტული თვისების გამომხატველია. სხვა სიტყვებით რომ ვთქვათ, ლაპარაკია არა ენათა კლასის, არამედ ტიურინგის მანქანების კონკრეტულ კლასზე.
ჩვენ იმის დამტკიცებაც შეგვიძლია, რომ ენათა კლასი, რომელიც ამ თვისებას გამოხატავს, არ არსებობს. ამისათვის საკმარისია ორი M_1 და M_2 ტიურინგის მანქანის პოვნა, რომლისთვისაც $L(M_1) = L(M_2)$, მაგრამ $\langle M_1 \rangle \in T_{TM}$ და $\langle M_2 \rangle \notin T_{TM}$.



ნახ. 11: $L(M_1) = L(M_2)$, მაგრამ $\langle M_1 \rangle \in T_{TM}$ და $\langle M_2 \rangle \notin T_{TM}$

სავარჯიშო 1.7: დაამტკიცეთ, რომ ნახ. **??**-ში მოყვანილი ტიურინგის მანქანებისთვის $L(M_1) = L(M_2)$, მაგრამ $\langle M_1\rangle \in T_{TM}$ და $\langle M_2\rangle \notin T_{TM}$.

სავარჯიშო 1.8: დაამტკიცეთ T_{TM} ენის არაამოხსნადობა მასზე შეჩერების ამოცანის დაყვანით.

მაგალითი 1.5: მოცემულია ორი ტიურინგის მანქანა M_1 და M_2. $L(M_1) = L(M_2)$?
არც აქ შეიძლება რაისის თეორემის გამოყენება იმ სახით, რომლითაც ის ზემოთ იქნა ჩამოყალიბებული, რადგან აქ ორ ენაზეა ლაპარაკი. ამ ამოცანის გადასაჭრელად რაისის თეორემის განვრცობაა საჭირო.

განმარტება 1.2: ტიურინგის მანქანების მიერ ამოცნობად ენათა წყვილების სიმრავლეს

$$S = \{(L_1, L_2) | L_1, L_2 \text{ ტიურინგის მანქანების მიერ ამოცნობადი ენებია}\}$$

ენათა წყვილების თვისება ეწოდება. S თვისება არატრივიალურია, თუ არსებობს ამოცნობად ენათა ისეთი ორი წყვილი (L_1, L_2) და (L_3, L_4), რომ $(L_1, L_2) \in S$ და $(L_3, L_4) \notin S$.

თეორემა 1.2: (რაისის გაფართოებული თეორემა ენათა წყვილებისთვის) თუ ენათა წყვილების თვისება S არატრივიალურია, მაშინ

$$L_S = \{\langle M_1, M_2\rangle \,|(L(M_1), L(M_2)) \in S\}$$

ენა არაამოხსნადია.

სავარჯიშო 1.9: დაამტკიცეთ ზემოთ მოყვანილი თეორემა.

სავარჯიშო 1.10: გამოიყენეთ ზემოთ ჩამოყალიბებული თეორემა იმის დასამტკიცებლად, რომ M_1 და M_2 ტიურინგის მანქანებისთვის $L(M_1) = L(M_2)$ დადგენა შეუძლებელია.

1.8 სირთულის P და NP კლასები

თუ მოცემულია რეკურსიული ფუნქცია $f : \Sigma^* \to \Sigma^*$, მაშინ უნდა არსებობდეს ერთი მაინც ტიურინგის მანქანა M, რომელიც ამ ფუნქციას გამოითვლის: $M(w)$ ნიშნავს, რომ w მონაცემით გაშვებულ M-ს გაჩერების შემდეგ მეხსიერების კვალზე ექნება დაწერილი $f(w)$. რა თქმა უნდა, სხვადასხვა სიგრძის w სიტყვისათვის M გამოთვლას სხვადასხვა დროს მოანდომებს, რომელსაც შემდეგნაირად აღვნიშნავთ: $T'(M(w))$. შეიძლება ისე აღმოჩნდეს, რომ $w_1 \neq w_2$ და $|w_1| = |w_2|$, მაგრამ $T'(M(w_1)) \neq T'(M(w_2))$. ამიტომ განიხილავენ შემთხვევებს, როდესაც

$$W_n = \{w_1, ..., w_k \in \Sigma^* \ : \ |w_i| = n, 1 \leq i \leq k\} \text{ და } T_M(n) = \max_{1\leq i\leq k} \{T'(w) \mid |w| = n\}.$$

ეს იმას ნიშნავს, რომ თუ ტიურინგის მანქანა M მონაცემად n სიგრძის სიტყვას მიიღებს, მისი მუშაობის დროდ ყველა შესაძლო შემთხვევიდან მაქსიმალურს იღებენ. $T_M(n)$ თავის მხრივ n ცვლადზე, ანუ მონაცემის სიგრძეზე დამოკიდებული ფუნქციაა. სირთულის თეორიაში აინტერესებთ ამ ფუნქციის ზრდის რიგი $O(T_M(n))$. ამითი კონკრეტული ტიურინგის მანქანის გამოთვლის სისწაფის ზრდის რიგი ფასდება.
თუ განვიხილავთ *ყველა* ტიურინგის მანქანას, რომელიც მოცემულ f ფუნქციას გამოითვლის, მაშინ შეიძლება ზოგადად ფუნქციის გამოთვლის სისწრაფის შეფასება: ქვედა ზღვრის გამოთვლა. თუ

$$L(f(n)) = \min_{1\leq i<\infty} \{T_{M_i}(n) \mid M_i \text{ ტიურინგის მანქანაა, რომელიც } f \text{ ფუნქციას გამოითვლის}\},$$

მაშინ $\Omega(L(f(n)))$ ამ ფუნქციის გამოთვლის ქვედა ზღვარს გვიჩვენებს.

ამრიგად, რაღაცა ამოცანის ქვედა ზღვარი იმას გვიჩვენებს, თუ რა სისწრაფის შეიძლება იყოს ამ ამოცანის გადასაჭრელად შემოთავაზებული *ყველა ზე სწრაფი* ალგორითმი, ანუ ამ ქვედა ზღვარზე უფრო სწრაფი ალგორითმი არ არსებობს. ზედა ზღვარი კი ამ ფუნქციის გამოსათვლელად შემოთავაზებული *კონკრეტული* ალგორითმის ბიჯების მაქსიმალურ რაოდენობას გვიჩვენებს.

როგორც ზემოთ აღინიშნა, თუ რაიმე ამოცანის ამოსახსნელად მოიძებნება დეტერმინისტული ტიურინგის მანქანა M, რომელიც პოლინომიურ დროში ამოხსნის ამ ამოცანას, ანუ $T_M(w) = O(|w|^c)$, $c = const$, ამ ამოცანას „პოლინომიურ დროში ამოხსნადს" უწოდებენ. პოლინომიურ დროში ამოხსნად ამოცანათა სიმრავლეს აღნიშნავენ P-თი (polynomial-time).
თუ რაიმე ამოცანის გადასაჭრელად არსებობს ისეთი არადეტერმინისტული მანქანა ND, რომელიც პოლინომიურ დროში ამოხსნის ამ ამოცანას, ანუ $T_N D(|w|) = O(|w|^c)$, $c = const$, ამ ამოცანას „არადეტერმინისტულად პოლინომიურ დროში ამოხსნადს" უწოდებენ. არადეტერმინისტულად პოლინომიურ დროში ამოხსნად ამოცანათა სიმრავლეს აღნიშნავენ სიმბოლოთი NP (non-deterministic polynomial-time).
ცხადია, რომ $P \subset NP$, რადგან დეტერმინისტული ტიურინგის მანქანები არადეტერმინისტულის კერძო შემთხვევაა, სადაც ყოველ ბიჯში მოქმედება ცალსახადაა განსაზღვრული რაიმე არჩევანის საშუალების გარეშე.
მაგრამ არაა გაცემული პასუხი ინფორმატიკის უმნიშვნელოვანეს შეკითხვაზე: $P = NP$? ანუ არსებობს თუ არა არადეტერმინისტული მანქანის გადეტერმინისტულობის ზოგადი მეთოდი, რომლის შედეგადაც მდგომარეობათა სიმრავლე ექსპონენციურად არ გაიზრდება?

იტყვიან, რომ ამოცანა A ამოცანა B-ზე დაიყვანება პოლინომიურ დროში ($A \leq_P B$), თუ არსებობს ორი ტიურინგის მანქანა M_1 (რომელიც A ამოცანას ხსნის) და M_2 (რომელიც B ამოცანას ხსნის) და პოლინომიურ დროში გამოთვლადი რეკურსიული ფუნქცია $\tau : \Sigma^* \to \Sigma^*$ ისეთი, რომ $w \in L(M_1) \Rightarrow \tau(w) \in L(M_2)$.
NP სიმრავლისათვის გამოყოფენ ე.წ. „რთულ" ამოცანათა კლასს, ანუ ისეთ ამოცანათა სიმრავლეს, რომლებზედაც ამ სიმრავლის ყველა დანარჩენი ამოცანა პოლინომიურ დროში დაიყვანება.
სხვა სიტყვებით რომ ვთქვათ, A ამოცანა NP-რთულია, თუ

$$\forall B \in NP, \;\; B \leq_P A.$$

თუ რაიმე NP რთული ამოცანა თვითონაც NP კლასს ეკუთვნის, მაშინ ამბობენ, რომ ეს ამოცანა NP-სრულია.

მნიშვნელოვანია ის ფაქტი, რომ ამოცანათა პოლინომიურ დროში დაყვანის პროცესი ტრანზიტულია:

$$A \leq_P B, \;\; B \leq_P C \;\; \Rightarrow \;\; A \leq_P C.$$

ეს კი იმას ნიშნავს, რომ თუ ერთი NP სრული ამოცანისთვის მაინც აღმოჩნდა პოლინომიურ დროში მომუშავე ალგორითმი, ასეთი ალგორითმი NP სიმრავლის ყველა ამოცანისთვის აღმოჩნდება და $P = NP$ --- ნებისმიერი ამოცანა „რეალურ" (პოლინომიურ) დროში ამოიხსნება. მაგრამ თუ აღმოჩნდება ისეთი NP სრული ამოცანა, რომელიც პოლინომიურ დროში ვერ ამოიხსნება (ანუ მისი ქვედა ზღვარი იქნება $\Omega(2^n)$), მაშინ დამტკიცდება $P \neq NP$ --- NP სიმრავლეში იარსებებს ისეთი ამოცანები, რომლებიც დეტერმინისტული მანქანით რეალურ დროში ვერ ამოიხსნება.

ათასობით NP სრული ამოცანის რამოდენიმე მაგალითია:

- ჰამილტონის ციკლი გრაფში
- მთელ რიცხვთა დაყოფა: მოცემულია მთელ რიცხვთა სიმრავლე A. შეიძლება თუ არა ეს სიმრავლე დავყოთ ორ დისკრეტულ ქვესიმრავლედ A_1, A_2 ($A_1 \cup A_2 = A$, $A_1 \cap A_2 = \emptyset$) ისე, რომ $\Sigma_{a \in A_1} a = \Sigma_{b \in A_2} b$?
- ზურგჩანთის ამოცანა
- ქვეგრაფის ამოცანა: მოცემულია ორი გრაფი. გაარკვიეთ, არის თუ არა ერთი გრაფი მეორეს ქვეგრაფი
- უმოკლესი მანძილი გრაფის წვეროებს შორის: მოცემულია შეწონილი გრაფი და მისი ორ წვერო. დაადგინეთ ამ ორ წვეროს შორის უმოკლესი გზა.
- შტაინერის ხე: მოცემულია გრაფი. დახაზე მინიმალური ხე, რომელიც ამ გრაფის ყველა წვეროს აერთებს (აღსანიშნავია, რომ შტაინერის ხეში დამატებითი კვანძებიც შეიძლება არსებობდეს -- ისეთები, რომლებიც არ გვხვდება მოცემულ გრაფში)
- გრაფის შეღებვა: მოცემულია გრაფი. დაადგინეთ, სულ ცოტა რამდენი ფერია საჭირო იმისათვის, რომ ამ გრაფის ყოველი წვერო შეიღებოს რაღაცა ფრად ისე, რომ ყოველ ორ წვეროს, რომელიც წიბოთია შეერთებული, სხვადასხვა ფერი ჰქონდეს

რა თქმა უნდა, არსებობს NP რთული ამოცანებიც, რომლებიც არაა NP სრული, ანუ არ შეიძლება მათი არადეტერმინისტული მანქანებით პოლინომურ დროში ამოხსნა (რაც იმას ნიშნავს, რომ არაა შესაძლებელი მათი შესაძლო

ამონახსნის პოლინომიურ დროში გადამოწმება), მაგრამ ყველა ამოცანა NP სიმრავლიდან მასზე დაიყვანება. ასეთი ამოცანის მაგალითია ტიურინგის მანქანათა შეჩერების ამოცანა, რომელიც ჩვენ ზემოთ განვიხილეთ: ყველა ამოცანა NP-დან მასზე დაიყვანება, მაგრამ თვითონ იგი ვერ იქნება NP სიმრავლეში, რადგან ამოუხსნადია.
ბუნებრივად ისმის შეკითხვა: არსებობს გამოთვლადი NP რთული ფუნქციები, ანუ ისეთები, რომელთა გამოთვლაც არადეტერმინისტული მანქანებით არ შეიძლება პოლინომიურ დროში, მაგრამ რომელზედაც ყველა დანარჩენი ფუნქცია NP სიმრავლიდან დაიყვანება? სხვა სიტყვებით რომ ვთქვათ, თუ მოცემულია გამოთვლადი ფუნქცია f, რომელზედაც პოლინომიურ დროში დაიყვანება ყველა ფუნქცია NP სიმრავლიდან და ორი სიტყვა w, v, იმის დადგენა, ჭეშმარიტია თუ არა $f(w) = v$, შეუძლებელია პოლინომიურ დროში.
ასეთი ფუნქციების სიმრავლე კონტინუუმის სიმძლავრისაა, მაგრამ პრაქტიკაში არც თუ ისე მრავლად არის ცნობილი. თუმცა მაგალითები მაინც არსებობს: მოცემულ თამაშში დაადგინეთ, მომგებიანია თუ არა რაღაცა კონკრეტული სტრატეგია? ყოველთვის ამის პოლინომიურ დროში გადამოწმება არაა შესაძლებელი.

1.9 თეორემა საშუალედო ენის არსებობის შესახებ

როგორც აქამდე ვნახეთ, NP სიმრავლეში გვხვდება „ადვილად", ანუ პოლინომიურ დროში ამოხსნადი ამოცანები და „რთული", ანუ NP სრული ამოცანები. თანმედროვე მეცნიერების ერთ-ერთი ყველაზე მნიშვნელოვანი ამოცანაა P vs. NP საკითხი, ანუ ემთხვევა თუ არა ეს ორი სიმრავლე ერთმანეთს. იმ შემთხვევაში, თუ $P = NP$, ნებისმიერი ამოცანა, რომლის პასუხის გადამოწმებაც (სერტიფიცირება) შესაძლებელი იქნება პოლინომიურ დროში, პოლინომიურ დროში ამოხსნადი იქნებოდა. მაგრამ თუ $P \neq NP$, მაშინ ორი შესაძლებლობა არსებობს: ან NP სიმრავლე ზუსტად ორ P და NPC კლასებად იყოფა, ან უნდა არსებობდეს ისეთი ენა (ამოცანა), რომელიც არ ამოიხსნება პოლინომიურ დროში და არც NP სრულია.
შემდეგი თეორემა სწორედ ამ ფაქტს ამტკიცებს.

თეორემა 1.3: თუ $P \neq NP$, მაშინ არსებობს ისეთი ენა NP სიმრავლეში, რომელიც არ ეკუთვნის P სიმრავლეს და ***ამავდროულად*** არ არის NP სრული.

დამტკიცება: როგორც ცნობილია, შესაძლებელია ყველა ისეთი $M_1, M_2, \ldots$ ტიურინგის მანქანის გადანომვრა, რომელთა მუშაობის დრო ზემოდან პოლინომიური ფუნქციითაა შემოსაზღვრული. ასევე შეიძლება $R_1, R_2, \ldots$ პოლინომიურ დროში დაყვანათა ალგორითმების გადანომრაც. რადგან ეს შესაძლებელია, უნდა არსებობდეს ტიურინგის მანქანა TM_1, რომელიც რიგ-რიგობით მოგვცემს ასეთ M_i მიმდევრობას; ანალოგიურად უნდა არსებობდეს TM_2 ტიურინგის მანქანაც, რომელიც რიგ-რიგობით მოგვცემს R_i მიმდევრობას.

შენიშვნა: აღსანიშნავია, რომ ამდაგვარი გადათვლის პოვნა ენათა ყველა კლასისთვის შესაძლებელი არაა. არსებობს ისეთ ენათა კლასიც, რომელთათვისაც გადათვლის პოვნა შესაძლებელი, მაგრამ ძალიან ძნელია.

ამას გარდა, TS იყოს ისეთი ტიურინგის მანქანა, რომელიც გადაჭრის SAT ამოცანას (რადგან დავუშვით, რომ $P \neq NP$, ეს მანქანა პასუხს ექსპონენციურ დროში გამოიანგარიშებს).
ახლა კი აღვწეროთ ისეთი L ენა, რომელიც არ იქნება P სიმრავლეში და არც NP სრულია. ასეთ ენას ჩვენ პირდაპირ ვერ აღვწერთ, ამიტომ მოვიყვანთ ისეთი ტიურინგის მანქანის მაგალითს, რომელიც მას გადაწყვეტს (მხოლოდ მის სიტყვებს მიიღებს), რითაც დავამტკიცებთ, რომ ასეთი ენა უნდა არსებობდეს.
თუ მოცემულია სპეციალური ფუნქცია $f : \mathbb{N} \to \mathbb{N}$ (რომლის განსაზღვრაც თეორემის დამტკიცების მთავარი ნაწილია), ზემოთ აღნიშნული ტიურინგის მანქანა TK შემდეგნაირად განისაზღვრება:

$TK(x)$: თუ $TS(x) =$ "yes" და ამავდროულად $f(x)$ ლუწია, მაშინ მიიღე x, წინააღმდეგ შემთხვევაში არ მიიღო x.

სხვა სიტყვებით რომ ვთქვათ, TK მხოლოდ მაშინ იღებს x სიტყვას, თუ x მოცემულია კონიუნქციური ნორმალური ფორმით, ცვლადების რომელიმე კომბინაციაზე იგი 1 ხდება (სრულდება) და მის სიგრძეზე გამოთვლილი f ფუნქცია ლუწია.
რაც შეეხება f ფუნქციას, იგი მონოტონურად ზრდადია, ანუ $f(n+1) \geq f(n)$. თანაც f ძალიან ნელა იზრდება.
განვსაზღვროთ ტიურინგის მანქანა TF, რომელიც ამ ფუნქციას გამოითვლის. პირველ რიგში აღვნიშნოთ, რომ მისი მონაცემი n კოდირებულია როგორც 1^n (n ცალი 1). TF ორ ეტაპად მუშაობს, თანაც ყოველი ეტაპი n ბიჯს გრძელდება.
შეიძლება ჩავთვალოთ, რომ ეს ტიურინგის მანქანა თავს ამოძრავებს მარჯვნივ, სანამ არ შეხვდება პირველი ცარიელი სიმბოლო და გზადაგზა რაღაცას ითვლის. შემდეგ - მეორე სტადიაზე - იგი თავს მარცხნივ ამოძრავებს, სანამ არ შეხვდება მეხსიერების დასაწყისის სიმბოლო $\triangleright$ და ასევე გზაში რაღაცას ითვლის.

პირველ ფაზაში TF ცდილობს მაქსიმალურად ბევრი $f(0), f(1), f(2), \ldots$ გამოანგარიშებას (რამდენსაც მოასწრებს n ბიჯში). დავუშვათ, რომ ბოლოს გამოთვლილი სიდიდე იყო $f(i) = k$. $f(n)$ სიდიდე იქნება ან k, ან $k+1$ იმის მიხედვით, თუ როგორ განვითარდება გამოთვლა მეორე ფაზაში.
მეორე ფაზაში მუშაობა იმაზეა დამოკიდებული, პირველ ფაზაში გამოთვლილი k რიცხვი კენტია თუ ლუწი. დავუშვათ, რომ $k = 2i$ ლუწია. მაშინ TF ითვლის $M_i(z)$, $S(z)$ და $TF(|z|)$, სადაც z რიგ-რიგობით გაირბენს ლექსიკოგრაფიულად დალაგებული Σ^* სიმრავლის ყველა ელემენტს და შეჩერდება მაშინ და მხოლოდ მაშინ, თუ აღმოაჩენს ისეთ z სიტყვას, რომ

$$TK(z) \neq M_i(z),$$

სადაც TK არის L ენის განმსაზღვრელი ტიურინგის მანქანა.
TK მანქანის განსაზღვრების თანახმად, იძებნება ისეთი z სიტყვა, რომლისთვისაც

- $M_i(z) =$„კი” და $S(z) =$„არა” ან $f(|z|)$ კენტია;
- $M_i(z) =$„არა” და $S(z) =$„კი” და $f(|z|)$ ლუწია.

აქაც გამოთვლა n ბიჯის შემდეგ წყდება და თუ ასეთი z არ აღმოჩნდა, $f(n) = k$. თუ ამ n ბიჯში ასეთი z აღმოჩნდა, მაშინ $f(n) = k + 1$.

თუ პირველ ფაზაში გამოთვლილი $k = 2i + 1$ კენტია, მაშინ მეორე ფაზა შემდეგნაირად ვითარდება: TF ითვლის $R_i(z)$, (z), $S(R_i(z))$ და $F(|T_i(z))|)$, სადაც z ასევე ლექსიკოგრაფიულად გარბის Σ^* სიმრავლის ყველა მნიშვნელობას. აქ უკვე TF ეძებს ისეთ z სიტყვას, რომლისთვისაც

$$K(R_i(z)) \neq S(z)$$

უნდა აღინიშნოს, რომ R_i დაყვანაა და, აქედან გამომდინარე, სიტყვას იძლევა.

TK მანქანის განსაზღვრების თანახმად, იძებნება ისეთი z სიტყვა, რომლისთვისაც

- $S(z) =$„კი” და ან $S(R_i(z)) =$„არა” ან $f(|z|)$ კენტია;
- $S(z) =$„არა” და $S(R_i(z)) =$„კი” და $f(|z|)$ ლუწია.

თუ გამოთვლისთვის გამოყოფილ n ბიჯში ასეთი z მოიძებნა, მაშინ $f(n) = k + 1$. წინააღმდეგ შემთხვევაში $f(n) = k$.

აღსანიშნავია, რომ TF მკაცრად განსაზღვრული მანქანაა და $O(n)$ ბიჯში გამოითვლის $f(n)$ ფუნქციას.
აქედან გამომდინარე, ასევე TK, რომელიც L ენას განსაზღვრავს, იქნება მკაცრად განსაზღვრული.
ადვილი საჩვენებელია, რომ $L \in NP$: ნებისმიერ x მონაცემზე ჩვენ ჯერ $SAT(x)$ და მერე $f(|x|)$ უნდა გამოვითვალოთ, რაც არადეტერმინისტულად პოლინომიურ დროშია შესაძლებელი.

ახლა დავუშვათ, რომ $L \in P$. მაშინ იარსებებს ისეთი M_i დასაწყისში ნახსენები გადანომვრის მიხედვით, რომლისთვისაც $TK(z) = M_i(z), \forall z$ (სხვა სიტყვებით რომ ვთქვათ, TK მანქანა ამ სიაში შეგვხვდება). მაგრამ ასეთ შემთხვევაში TF მანქანის მეორე ფაზაში არ აღმოჩნდება ისეთი z, რომლისთვისაც $TK(z) \neq M_i(z)$ და $f(n)$ ფუნქცია აღარ გაიზრდება და $f(n) = 2i, \forall n > n_0$. ეს კი იმას ნიშნავს, რომ დაწყებული რაღაც ადგილიდან L ენა „გადაიზრდება” SAT ამოცანაში და, აქედან გამომდინარე, იქნება NP სრული, რაც მას $P \neq NP$ დაშვებით P სიმრავლიდან გამორიცხავს.

ახლა დავუშვათ, რომ $L \in NPC$. მაშინ უნდა არსებობდეს დაყვანა R_i SAT ამოცანიდან L ენაზე, რაც იმას ნიშნავს, რომ $TK(R_i(z)) = S(z), \forall z$. აქედან გამომდინარე, TF მანქანის მეორე ფაზაში $k = 2i+1$ შემთხვევისათვის შესაბამისი z არ მოიძებნება, რის შედეგადაც დაწყებული რაღაც ადგილიდან $f(n)$ არ გაიზრდება და დარჩება კენტი. აქედან გამომდინარე, L ენა სასრული გამოვა, ანუ $L \notin NPC$.

ორივე დაშვებას ($L \in P$ და $L \in NPC$) წინააღმდეგობამდე მივყავართ. აქედან გამომდინარე, L „საშუალედო” უნდა იყოს, ანუ $L \notin P$ და $L \in NPC$.

რ.დ.გ.

<u>სავარჯიშო 1.11:</u> დაამტკიცეთ, რომ ნებისმიერი სასრული ენა P სიმრავლეს ეკუთვნის.

<u>სავარჯიშო 1.12:</u> ზედა თეორემის დამტკიცების რომელ ადგილებში გამოვიყენეთ $P \neq NP$?

$P \neq NP$ შემთხვევაში შუალედური ენების არსებობა ძალიან მნიშვნელოვანია პრაქტიკაშიც. მაგალითად, ნატურალურ რიცხვთა ფაქტორიზაციის ან გრაფთა იზომორფიზმის ამოცნისთვის ჯერ-ჯერობით ვერავინ მოიფიქრა პოლინომიური ალგორითმი და ვერც იმისი NP სრულობის დამტკიცება შეძლო. ამან გააჩინა საფუძვლიანი ეჭვი, რომ ეს ორი ამოცანა სწორედ ასეთ საშუალედო კლასს უნდა ეკუთვნოდეს, მაგრამ ეს მხოლოდ იმ პირობით, რომ $P \neq NP$.

1.10 სირთულის თეორიიდან გამოტანილი დასკვნები

როგორც ვნახეთ, ამოცანები რამოდენიმე ძირითად კლასად შეიძლება დაიყოს: გამოთვლადი და არაგამოთვლადი, P და NP. არსებობს კიდევ მრავალი ამოცანათა კლასი, მაგალითად ისეთის, რომლის ამოხსნაც (გაპარალელების შედეგად) ლოგარითმულ დროში $O(\log n)$ შეიძლება (დახარისხება, არითმეტიკული ოპერაციები და ა.შ.).
ასეთ ამოცანათა კლასს „ნიკის კლასს" უწოდებენ და აღნიშნავენ როგორც NC. დღეისათვის ერთ-ერთი ძალიან მნიშვნელოვანი საკითხია, შესაძლებელია თუ არა P კლასში მყოფი ყველა ამოცანის ისე გაპარალელება, რომ მისი გამოთვლის დრო პოლინომიურიდან პოლი-ლოგარითმულზე დავიდეს, ანუ ზედა ზღვარი გახდეს $O(\log^k n)$ გავრცელებულია ეჭვი, რომ ორი ნატურალური რიცხვის უდიდესი საერთო გამყოფის ამოცანა არ ეკუთვნის NC კლასს: ევკლიდეს ალგორითმის პრინციპული გაუმჯობესება უკვე რამოდენიმე ათასი წელია ვერ მოხერხდა. აქედან გამომდინარე, ევკლიდეს ალგორითმის შესწავლა სირთულის თეორიის თვალსაზრისით ძალიან საინტერესო უნდა იყოს.

ბუნებრივად ისმის შეკითხვა, რაში გვეხმარება სირთულის თეორია? რას გვმატებს არადეტერმინისტული ავტომატების შექმნა, რომლის არამც თუ პრაქტიკული რეალიზაცია, არამედ გამოთვლის პროცესის გაგებაც კი შეუძლებელია?
ამ შეკითხვაზე შესაძლო პასუხი მათემატიკური ანალიზის პარალელით შეიძლება: რაში გვჭირდება ნამდვილ რიცხვთა სიმრავლე, თუ მის უდიდეს ნაწილს საერთოდ ვერც კი წარმოვიდგენთ? მაგრამ ამ რიცხვთა შემოღებამ საგრძნობლად გააადვილა კალკულუსის და, საერთოდ, მათემატიკის განვითარება, თეორემათა მტკიცება და რეალურ ცხოვრებაში არსებული პრობლემების კლასიფიკაცია. როგორც ცნობილია, ანალიზის აგება ტრანსცენდენტულ რიცხვთა გარეშეც შეიძლება, მაგრამ ამ შემთხვევაში თეორემათა მტკიცება წარმოუდგენლად ძნელდება. ასეა ინფორმატიკაშიც: არადეტერმინისტულ მანქანათა ჰიპოთეტური მოდელის შემოტანა თეორემათა მტკიცებაში, ამოცანათა კლასიფიკაციაში და სათანადო დასკვნების გამოტანაში გვიწყობს ხელს.

თუ გვხვდება ახალი ამოცანა, პირველ რიგში უნდა დავადგინოთ, შეიძლება თუ არა მისი ალგორითმულად ამოხსნა. თუ ამოხსნადია, უნდა დავადგინოთ, არის თუ არა იგი NP სრული. რადგან NP სრული ამოცანებისათვის არაა ცნობილი ეფექტური ამონახსნი და თეორიული შედეგებიც იმაზე მიგვითითებს, რომ ასეთი ალგორითმის პოვნა პრაქტიკაში ძალიან ძნელი ან შეუძლებელი უნდა იყოს, ამიტომ არ ეძებენ ასეთი ამოცანების ზუსტ ალგორითმებს (უხეშად რომ ვთქვათ, ძებნაში დროს არ კარგავენ). NP სრული ამოცანებისათვის სწრაფი ალგორითმების პოვნის უიმედობას აძლიერებს შემდეგი შედეგი:

არ არსებობს ისეთი პოლინომიური ალგორითმი, რომელიც ნებისმიერი არადეტერმინისტული სასრული ავტომატისათვის მის ექვივალენტურ მინიმალური მდგომარეობების მქონე დეტერმინისტულ სასრულ ავტომატს შეადგენს.

ერთად-ერთი იმედი არის ის, რომ ეს თეორემა ერთი რაიმე უნივერსალური მეთოდის არსებობას გამორიცხავს. ცალკეულ შემთხვევებში არაა გამორიცხული ოპტიმალური ზომის დეტერმინისტული სასრული ავტომატის შედგენა და თუ ეს ზომა პოლინომიური აღმოჩნდა, $P = NP$ დამტკიცება.

ალგორითმების შედგენისას უნდა გავითვალისწინოთ სამი ასპექტი (რა თქმა უნდა, ყველა ეს საკითხი აღარ იქნება აქტუალური, თუ $P = NP$):

1. სისწრაფე: შედგენილი ალგორითმი ჩვენთვის დამაკმაყოფილებელი სისწრაფით უნდა მუშაობდეს
2. მონაცემთა სისრულე: ალგორითმი უნდა ითვლიდეს შედეგს მთელს განსაზღვრის არეზე
3. სიზუსტე: ყოველ მონაცემზე ალგორითმი ზუსტად იმ შედეგს უნდა იძლეოდეს, როგორიც მას მოეთხოვება

დღეისათვის არაა ცნობილი, შეიძლება თუ არა NP სრული ამოცანებისათვის სამივე პუნქტის ერთდროულად შესრულება ($P = NP$ პრობლემა). ამიტომ ასეთი ამოცანებისათვის ეძებენ ისეთ ალგორითმებს, რომლებიც ერთ-ერთ პუნქტს არ ასრულებს:

პირველ რიგში უნდა განვსაზღვროთ, საჭიროა თუ არა ამოცანის ამოხსნა ყველა იმ მონაცემზე, რაც თეორიულად მოეთხოვება -- ხშირად თეორიაში მოთხოვნილი ბევრი მონაცემი პრაქტიკაში სრულებით არ გვხვდება. მაგალითად, გრაფის შეღებვის ამოცანა ადვილად გადაიჭრება, თუ არ განვიხილავთ ნებისმიერ გრაფს და მხოლოდ პლანარულებით შემოვიფარგლებით: ამ შემთხვევაში საკმარისია 4 ფერი. ამას გარდა, არაორიენტირებულ გრაფებზე დასმული თითქმის ყველა ამოცანა ეფექტურად გადაიჭრება, თუ ნებისმიერი გრაფის მაგივრად განვიხილავთ ხეებს (აციკლურ გრაფებს). ამითი აიხსნება მატროიდებზე აგებულ ალგორითმთა ეფექტურობაც.

მიახლოებითი ალგორითმები: ყოველ მონაცემზე მიღებული შედეგი ზუსტი შედეგის δ სიახლოვეშია. ამის მაგალითად გრაფის წიბოებით გადაფარვის ზემოთ ნახსენები ამოცანის ამოხსნა შეიძლება მოვიყვანოთ. მოცემული გრაფისათვის შემთხვევითი პრინციპით აირჩიე ერთი წიბო, ჩართე მისი ორივე წვერო საძიებელ სიმრავლეში და გრაფში წაშალე ყველა ის წიბო, რომელიც დაკავშირებულია ამ წვეროებთან. თუ ამ ოპერაციების შემდეგ მიღებული გრაფი ცარიელია, ალგორითმი დაასრულე. თუ არა და იგივე გაიმეორე.
ადვილი საჩვენებელია, რომ ამ ალგორითმის გაშვების შემდეგ მიღებული შედეგი მაქსიმუმ ორჯერ თუ აჭარბებს ოპტიმალურ შედეგს (დღეისათვის ეს ყველაზე ეფექტური ალგორითმია ამ ამოცანისათვის).

განმარტება 1.3: მოცემულია რაღაც ამოცანა, რომლის x მონაცემზე ოპტიმალური ამოხსნაა $Opt(x) \in \mathbb{R}$ და ალგორითმი A, რომლის შედეგი x მონაცემზე არის $A(x) \in \mathbb{R}$. იტყვიან, რომ ამ ალგორითმის შედეგი ამოცანის ოპტიმალური შედეგის $0 < \delta \leq 1$ სიახლოვეშია, თუ სრულდება შემდეგი უტოლობა:

$$\frac{|A(x) - Opt(x)|}{max\{A(x), Opt(x)\}} \leq \delta.$$

თუ რაიმე ამოცანას მოეძებნება ისეთი ალგორითმი, რომელიც ოპტიმალური შედეგის δ სიახლოვეშია, მას δ-მიახლოებადი ამოცანა ეწოდება.

ასეთი განსაზღვრებით გრაფების გადაფარვის ამოცანის ზემოთ მოყვანილი ალგორითმი ოპტიმალური შედეგის $\delta = \frac{1}{2}$ სიახლოვეშია.

გადაფარვის ამოცანა ე.წ. „ნაწილობრივ მიახლოვებად" ამოცანათა კლასში შედის: ისეთი ამოცანებისა, რომლებსაც ნებისმიერი სიზუსტით ვერ მივუახლოვდებით, მაგრამ გარკვეული სიზუსტით -- კი.

განმარტება 1.4: ამოცანა ნაწილობრივ მიახლოებადია, თუ წინასწარ ფიქსირებული $0 < \delta \leq 1$ რიცხვისთვის არსებობს δ-მიახლოებადი ალგორითმი.

არსებობს აგრეთვე ამოცანები, რომლისთვისაც შეიძლება ერთი რაიმე მიახლოებითი ალგორითმის დაწერა, რომელიც ნებისმიერი წინასწარ ფიქსირებული δ სიზუსტით მიუახლოვდება ოპტიმალურ შედეგს. ასეთ ამოცანებს სრულად მიახლოვებადი ეწოდება. სხვა სიტყვებით რომ ვთქვათ, დაიწერება ალგორითმი, რომელიც მონაცემად როგორც საწყისი ამოცანის მონაცემს, ასევე δ სიზუსტეს მიიღებს და შესაბამისი სიზუსტის პასუხს გამოითვლის.

განმარტება 1.5: ამოცანა სრულად მიახლოებადია, თუ არსებობს ალგორითმი $B(x, \delta)$, რომელიც ყოველ მონაცემზე ამ ამოცანის ოპტიმალური ამოხსნის δ სიახლოვეში მყოფ პასუხს გამოითვლის.

ასეთი სრულად მიახლოებადი ამოცანის მაგალითია ზურგჩანთის ამოცანა.

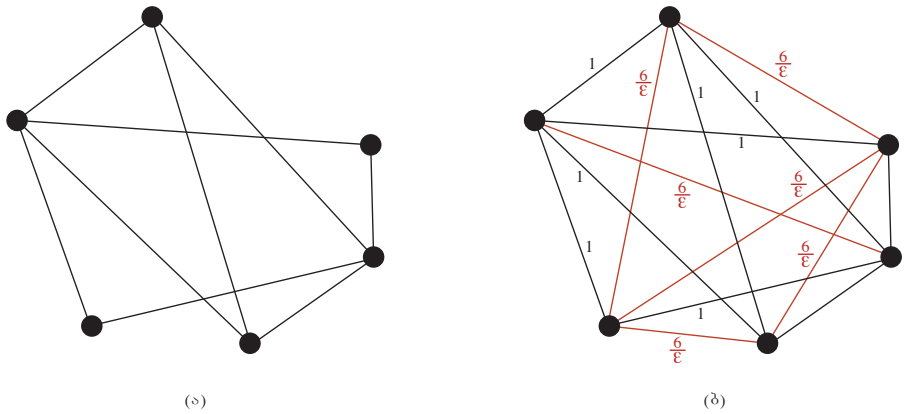
1.10.1 თეორემა TSP ამოცანის არამიახლოებადობის შესახებ და მისგან გამომდინარე შედეგები

დამტკიცებულია, რომ TSP ამოცანაში ასეთი მიახლოებები შეუძლებელია: ნებისმიერი $\delta > 0$ რიცხვისათვის მოიძებნება ისეთი კონკრეტული მონაცემი, რომლისთვისაც ნებისმიერი მიახლოებითი ალგორითმის შედეგი ზუსტი შედეგის δ მიდამოში არ იქნება, მაგრამ ამ ამოცანის კერძო შემთხვევების განხილვა ვითარებას ცვლის: თუ დამატებით მოვითხოვთ, რომ მოცემულ გრაფში სამკუთხედის უტოლობა სრულდებოდეს, ასეთი ამოცანა ნაწილობრივ მიახლოვებადი ხდება. თუ კიდევ დავადებთ შეზღუდვას, რომ გრაფი იყოს მოცემული ევკლიდურ სიბრტყეზე, მაშინ იგი სრულად მიახლოვებადი ხდება.

თეორემა 1.4: თუ $P \neq NP$, მაშინ TSP არ არის მიახლოებადი პოლინომიურ დროში.

დამტკიცება: დავამტკიცოთ, რომ ამოცანა „გამოიანგარიშეთ TSP ϵ მიახლოებით" NP სრულია, რაც $P \neq NP$ პირობით თეორემას დაამტკიცებს.

ამ ამოცანაზე დავიყვანოთ ჰამილტონის ციკლის ამოცანა. თუ მოცემულია ნებისმიერი გრაფი $G = \{V, E\}$, შევადგინოთ შეწონილი გრაფი $G' = \{V', E', w\}$, რომელშიც $V' = V$, $u, v \in V' \Rightarrow (u, v) \in E'$ და $w(u, v) = 1$, თუ $(u, v) \in E$, $w(u, v) = \frac{|V|}{\epsilon}$, თუ $(u, v) \notin E$ (ნახ. 12).



ნახ. 12: HC და TSP ამოცანების გრაფები

ცხადია, თუ G გრაფში არსებობს ჰამილტონის ციკლი, G' გრაფზე ϵ სიზუსტით ამოხსნილი TSP მოგვცემს პასუხს A, რომელიც $|V|$ რიცხვის ϵ მიდამოში იქნება: $\frac{A-|V|}{A} \leq \epsilon$.
თუ G გრაფში ჰამილტონის ციკლი არ არსებობს, მაშინ ოპტიმალური მარშრუტი $Opt \geq |V| + \frac{|V|}{\epsilon}$. აქედან გამომდინარე, მისი ϵ სიზუსტით გამოთვლილი პასუხი $A \geq |V| + \frac{|V|}{\epsilon}$, რაც $|V|$ ამონახსნის ϵ მიდამოში ვეღარ იქნება.

ესე იგი, თუ G' გრაფზე ამოვხსნით TSP ამოცანას ϵ სიზუსტით, ცალსახათ შევძლებთ იმის დადგენას, არსებობს თუ არა ჰამილტონის ციკლი G გრაფში და TSP ამოცანის ϵ სიზუსტით ამოხსნა NP სრულის გამოდის.

რ.დ.გ.

ამრიგად, NP სრული ამოცანები სამ კლასად შეიძლება დაიყოს:

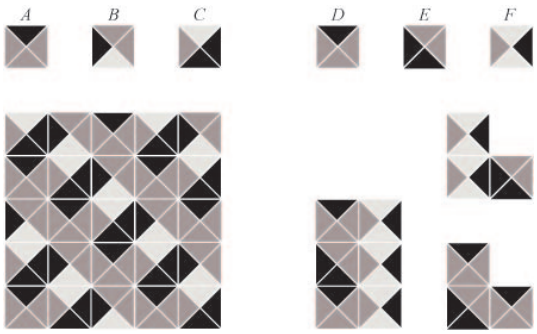
1. სრულად მიახლოვებად ამოცანათა კლასი --- სამწუხაროდ ძალიან ვიწრო. მასში შედის, მაგალითად, დავალებათა შესრულების ამოცანა ორ პროცესორზე

2. ნაწილობრივ მიახლოვებად ამოცანათა კლასი --- აგრეთვე მცირე, მაგრამ პირველზე უფრო ფართო, რომელშიც, მაგალითად, გრაფთა გადაფარვის ამოცანა შედის

3. არამიახლოვებად ამოცანათა კლასი --- ყველაზე ფართო კლასი, რომელშიც შედის TSP, გრაფებში დამოუკიდებელ წვეროთა სიმრავლის ამოცანა, სრული ქვეგრაფის გამოყოფის ამოცანა და სხვა.

აღსანიშნავია ის გარემოება, რომ არაამოხსნად ამოცანებშიც შეიძლება მონაცემთა სიმრავლის ან პირობის ისე შეზღუდვა, რომ იგი ამოხსნადად გადაიქცეს (როგორც, მაგალითად, TSP ამოცანაში მონაცემთა შეზღუდვისას იგი ნაწილობრივ მიახლოვებადი ან სრულად მიახლოვებადიც კი ხდებოდა).
საინტერესოა, რომ ასეთ შემთხვევებში შეცვლილი ამოცანები ხშირად NP სრული ხდება. ამის საილუსტრაციოდ განვიხილოთ „ფართობის დაფარვის" ამოცანა:

მოცემულია n სხვადასხვა ტიპის ერთნაირი ზომის კვადრტული ფიგურა $a_1, a_2, ..., a_n$ (თითო ტიპის ფიგურის რაოდენობა უსასრულოა). მოცემულია აგრეთვე წესები, თუ რა ტიპის კვადრატების დადება შეიძლება ერთმანეთის გვერდით და რა ტიპისა -- ერთმანეთის ზემოთ და ქვემოთ. ამას გარდა, ფიგურათა დატრიალება არ შეიძლება. დაადგინეთ, შეიძლება თუ არა მოცემული კვადრატებითა და წესებით ნებისმიერად დიდი ფართობის მქონე კვადრატის აგება, თუ ამ კვადრატის მარცხენა ქვედა კუთხეში განთავსებული კვადრატული ფიგურის ტიპი წინასწარაა მოცემული. ნახ. 13-ში მარცხნივ ნაჩვენებია ამ ამოცანის ერთი მაგალითი: მოყვანილია სამი ტიპის კვადრატული ფიგურა A, B, C, რომელთა გვერდები რადაცა ფერადაა შეღებილი. ორი ფიგურის ერთმანეთზე მიდება შეიძლება მაშინ და მხოლოდ მაშინ, თუ ამ ფიგურების შესაბამის გვერდებზე ფერები ერთნაირია.
ადვილი დასამტკიცებელია, რომ A, B, C ფიგურებითა და A ფიგურის მარცხენა ქვედა კუთხეში დადებით ნებისმიერი ფართობის კვადრატის აგება შეიძლება. მაგრამ თუ ავიღებთ D, E, F ფიგურებს, მაშინ შეიძლება მხოლოდ 2×2 კვადრატის აგება, ისიც იმ პირობით, თუ მარცხენა ქვედა კუთხეში D ფიგურას დავდებთ.



ნახ. 13: ფართობის დაფარვის ამოცანა

დამტკიცებულია, რომ ამ ამოცანისათვის ალგორითმული ამოხსნა არ არსებობს. მაგრამ თუ პირობას შევცვლით და ვიკითხავთ, შეიძლება თუ არა რაღაცა $k \times k$ ($k \in \mathbb{N}$) კვადრატის დაფარვა (უსასრულოდ დიდის მაგივრად), მაშინ ეს ამოცანა *NP* სრული ხდება.

1.11 არაგამოთვლად ფუნქციათა პრაქტიკაში გამოყენების მაგალითი

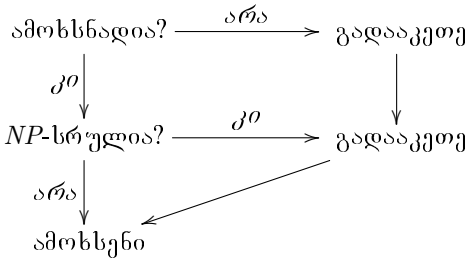
ყოველი ზემოთ თქმულის შემდეგ შეიძლება გაგვიჩნდეს ლოგიკური შეკითხვა: როგორ შეიძლება კონკრეტული თეორიული შედეგების პრაქტიკაში გამოყენება? ერთ-ერთი ასეთი გამოყენების სადემონსტრაციოდ განვიხილოთ ე.წ. „საქმიანი თახვის" ამოცანა.
როგორც ცნობილია, თახვები უაღრესად ბეჯითი ცხოველები არიან: გადადიან ტყეში, იღებენ მორებს, მდინარეში მიცურავენ და წყალსაგუბებლებს აშენებენ. ასეთ იქით-აქეთ სირბილს თახვი თავს არ ანებებს, სანამ თავის სამუშაოს ბოლომდე არ დაასრულებს. თახვების მსგავსად მოქმედებს ტიურინგის მანქანაც: გამოთვლის პროცესში თავაკი იქით-აქეთ დარბის და მეხსიერების კვალზე გარკვეულ სიმბოლოებს წერს.
1962 წელს უნგრელმა მათემატიკოსმა ტიბორ რადომ წამოაყენა „საქმიანი თახვის" ამოცანა: მოცემულია n მდგომარეობიანი ტიურინგის მანქანა ანბანით $\{0, 1\}$, რომელიც მუშაობას ცარიელი მეხსიერების კვალით იწყებს და როდესღაც ჩერდება. რამდენი 1 იქნება დაწერილი მეხსიერების კვალზე მანქანის გაჩერების შემდეგ? და ზოგადად: მაქსიმუმ რამდენი 1 შეიძლება დაწეროს n მდგომარეობის მქონე ტიურინგის მანქანამ გაჩერებამდე, თუ იგი მუშაობას ცარიელი მეხსიერებით დაიწყებს? ამ რიცხვს გამოსახავენ $\Sigma(n)$ ფუნქციით, რომელსაც რადოს ფუნქციას უწოდებენ. ანალოგიურად შეიძლება დავსვათ შეკითხვა: მაქსიმუმ რამდენი ბიჯის შემდეგ გაჩერდება n მდგომარეობის მქონე ტიურინგის მანქანა? მდგომარეობის ეს რიცხვი აღინიშნება ფუნქციით $S(n)$ (თუ ტიურინგის მანქანა არ შეჩერდება, მაშინ $S(n) = 0$). როგორც დამტკიცებულია, ეს ორივე ფუნქცია უფრო სწრაფად იზრდება, ვიდრე *ნებისმიერი* გამოთვლადი ფუნქცია, ანუ ორივე არაგამოთვლადია:

n	1	2	3	4	5	6
$\Sigma(n)$	1	4	6	13	≥ 4098	$\geq 10^{1439}$
$S(n)$	1	6	21	107	$\geq 47.176.870$	$\geq 10^{2879}$

მაგრამ, ამის და მიუხედავად, უაღრესად დიდი მნიშვნელობა აქვს მათ დათვლას კონკრეტული n-თვის. მაგალითად, თუ შევადგენთ ზემოთ ნახსენებ ტიურინგის მანქანას, რომელიც რიგ-რიგობით გადაამოწმებს ნატურალურ რიცხვებს და გაჩერდება მაშინ და მხოლოდ მაშინ, თუ ისეთ ლუწ რიცხვს აღმოაჩენს, რომელიც არ წარმოდგება ორი მარტივი რიცხვის ჯამის სახით, მაშინ შეიძლება გოლდბახის ჰიპოთეზის შემოწმება: დავითვლით ამ მანქანის მდგომარეობათა რაოდენობას n, გამოვითვლით $S(n)$ ფუნქციას და ამ მანქანას ვამუშავებთ $S(n)$ ბიჯის განმავლობაში. თუ იგი ამ დროში არ შეჩერდა, ესე იგი აღარასდროს შეჩერდება და ჩვენ რიცხვთა თეორიის ერთ-ერთ უმნიშვნელოვანეს საკითხს გადავჭრით.
მაგრამ, სამწუხაროდ, ჯერ-ჯერობით ასეთი პერსპექტივა საკმაოდ ბუნდოვანია.

1.12 მოკლე დასკვნა ალგორითმების თეორიის თვალსაზრისით

ახალი ამოცანის დასმისას უნდა გავარკვიოთ, შეიძლება თუ არა მისი ალგორითმულად გადაჭრა. თუ ასეთი რამ შეუძლებელია, უნდა განვსაზღვროთ, რა დამატებითი შეზღუდვაა საჭირო მონაცემებსა და პირობაზე, რომ იგი ამოხსნადი გახდეს.



აღსანიშნავია, რომ ხშირ შემთხვევაში არაამოხსნადი ამოცანის გადაკეთების შემდეგ იგი *NP* სრული ხდება. დღევანდელი მიდგომით, თუ რაიმე ამოცანაზე დამტკიცდა, რომ იგი *NP* სრულია, მის ზუსტ ამოხსნაზე დროს აღარ კარგავენ და გადაჭრის ალტერნატიულ გზებს ეძებენ: უნდა განისაზღვროს, რა დამატებითი შეზღუდვაა საჭირო მონაცემებსა და პირობაზე, რომ იგი პოლინომურ დროში ამოხსნადი გახდეს, ან როგორ შეიძლება მისი ამონახსნის რაღაცა δ სიზუსტით გამოთვლა -- არის თუ არა იგი სრულად მიახლოვებადი, ნაწილობრივ მიახლოვებადი ან არამიახლოვებადი. ამას გარდა, შესაძლებელია ამოცანათა ალბათური მეთოდებით გადაჭრა: ისეთი ალგორითმის დაწერა, რომ პასუხი ***ხშირ შემთხვევაში*** სწორი იყოს. რა თქმა უნდა, ასე მიღებულ პასუხზე დარწმუნებით ვერ ვიტყვით, რიმ დასმული ამოცანის ამოხსნას გვაძლევს, მაგრამ თუ მას ბევრჯერ გავუშვებთ და მიღებული პასუხებიდან ისეთს ავარჩევთ, რომელიც ყველაზე ხშირად გვხვდება, დიდი ალბათობით შეგვიძლია ვივარაუდოთ, რომ ეს ზუსტ ამოხსნას დაემთხვევა.

ამდაგვარი „ევრისტიკების" შედგენასა და ამოცანათა კლასიფიკაციაში გვეხმარება სირთულის თეორია, რომლის გარეშეც ასეთი რამ შეუძლებელი იქნებოდა.

2 სირთულის თეორიის გამოყენების მაგალითი: გიოდელის თეორემა არითმეტიკის არასრულობის შესახებ

2.1 დამტკიცების სისტემები

ცნობილი ავსტრიელი მეცნიერის კურტ გიოდელის (Kurt Gödel) მიერ მეოცე საუკუნის 30ან წლებში დამტკიცებული თეორემები თამამად შეიძლება მივიჩნიოთ კაცობრიობის აზროვნებაში შეტანილ გარდატეხად. თუ კლასიკური პერიოდიდან მოყოლებული მეცნიერება ნებისმიერი გამონათქვამის ლოგიკური მსჯელობით დამტკიცებასა ან უარყოფას გულისხმობდა, აღმოჩნდა, რომ ადამიანი მის მიერვე შექმნილ სისტემებშიც კი ვერ შეძლებს ბოლომდე გარკვევას: არითმეტიკაში არსებობს ისეთი გამონათქვამები, რომელთა არც დამტკიცება და არც უარყოფა არ შეიძლება. მათემატიკა არასრულყოფილი აღმოჩნდა.
წინამდებარე ტექსტი გიოდელის თეორემების მტკიცების საინტერესო ვარიანტს აღწერს, რომელიც გამომთვლელი მანქანების (ტიურინგის მანქანების) თეორიას ეყრდნობა.
ტექსტი ეფუძნება ცნობილი გერმანელი მეცნიერის ვოლფგანგ პაულის სახელმძღვანელოს Prof. Dr. Wolfgang J. Paul, Komplexitätstheorie, Teubner Verlag, 1978. დიდ მადლობას ვუხდით პროფ. როლანდ ომანაძეს კორექციისა და კრიტიკული შენიშვნებისთვის.

2.2 დამტკიცების სისტემები

განმარტება 2.6: დამტკიცების სისტემა აღიწერება $\mathcal{W} = (\Sigma, L, A, S)$ ოთხეულით, სადაც

- Σ სასრული ანბანია;
- $L \subset \Sigma^*$ გამოთვლადი სიმრავლეა, რომელსაც $\mathcal{W}$ სისტემის ენა ეწოდება;
- ყოველი $w \in L$ სიტყვა რაიმე მათემატიკური ობიექტის შესახებ გამონათქვამის ცალსახად აღმწერი უნდა იყოს;
- $A \subset L$ გამოთვლადი სიმრავლეა, რომელსაც $\mathcal{W}$ სისტემის აქსიომების სიმრავლე ეწოდება;
- S წარმოადგენს გამოთვლად სიმრავლეს, რომელსაც „დასკვნის კანონების" სიმრავლე ეწოდება და შემდეგი ტიპის სიტყვებისაგან შედგება:
  $w_1\#w_2\#...\#w_r$, სადაც $w_i \in L$ და $\# \notin L$.
  თუ $w_1\#w_2\#...\#w_r \in S$, მაშინ ამბობენ, რომ w_r პირდაპირ გამომდინარეობს $w_1, w_2, ..., w_{r-1}$-დან.

განმარტება 2.7: $\mathcal{W} = (\Sigma, L, A, S)$ დამტკიცების სისტემაში w_s გამონათქვამის ***დამტკიცება*** ეწოდება მიმდევრობას $b = w_1\#...\#w_s$, თუ ყველა $i \leq s$ ინდექსისთვის ჭეშმარიტია: $w_i \in A$ ან $\exists i_1, ..., i_r < i$ ისეთი, რომ $w_{i_1}\#...\#w_{i_r}\#w_i \in S$.

თუ არსებობს w გამონათქვამის დამტკიცება $\mathcal{W}$ სისტემაში, მაშინ ამბობენ, რომ w ***გამოყვანადია*** $\mathcal{W}$ სისტემაში. ცხადია, რადგან A და S გამოთვლადია, ასევე შესაძლებელი იქნება ისეთი ალგორითმის აგება, რომელიც ნებისმიერად მოცემული $b \in \{\Sigma \cup \{\#\}\}^*$ სიტყვისთვის განსაზღვრავს, არის თუ თუ არა b დამტკიცება $\mathcal{W}$ სისტემაში.

აქედან გამომდინარე, მივიღეთ

ლემა 2.1: თუ $\mathcal{W}$ დამტკიცების სისტემაა, $\{b : b$ დამტკიცებაა $\mathcal{W}$ სისტემაში$\}$ სიმრავლე გამოთვლადია.

სავარჯიშო 2.13: დაამტკიცეთ ზემოთ მოყვანილი ლემა.

შედეგი: ნებისმიერი $\mathcal{W}$ დამტკიცების სისტემისათვის სიმრავლე $\{w : w$ გამოყვანადია $\mathcal{W}$ სისტემაში$\}$ რეკურსიულად გადათვლადია.

დამტკიცება: w_0 იყოს $\mathcal{W}$ სისტემაში რაიმე გამოყვანადი გამონათქვამი. M იყოს ისეთი ტიურინგის მანქანა, რომელიც b მონაცემისთვის ამოწმებს, არის თუ არა იგი დამტკიცება მოცემულ $\mathcal{W}$ სისტემაში. თუ ეს ასეა, M ბეჭდავს b-ს მიერ დამტკიცებულ გამონათქვამს, წინააღმდეგ შემთხვევაში კი დაბეჭდავს w_0 გამონათქვამს. ცხადია,

$\{w : w$ გამოყვანადია $\mathcal{W}$ სისტემაში$\} = \{f_M(x) : x \in \{\Sigma \cup \{\#\}\}^*\}$

რ.დ.გ.

2.2.1 დამტკიცების სისტემის მაგალითი

მაგალითისათვის მოვიყვანოთ ელემენტარული რიცხვთა თეორიის დამტკიცების სისტემა.

$Z_E = (\Sigma_E, L_E, A_E, S_E)$, სადაც

$\Sigma_E = \{0, 1, x, (,), t, \cdot, =, \vee, \wedge, \neg, \rightarrow, \forall, \exists\}$,

$V = \{xw : w \in \{0,1\}^+\}$ ცვლადების სიმრავლეა.

განმარტება 2.8: ტერმების სიმრავლე ეწოდება ისეთ უმცირეს T სიმრავლეს, რომლისთვისაც ჭეშმარიტია:

1. $0 \in T$, $1 \in T$, $V \subset T$;
2. თუ $a, b \in T$ ტერმია, ასევე ტერმი იქნება $(a + b)$ და $(a \cdot b)$.

მაგალითად, ტერმია $((x10 + 1) \cdot 1)$.

განმარტება 2.9: არითმეტიკულ პრედიკატთა სიმრავლე ისეთ უმცირეს P სიმრავლეს ეწოდება, რომლისთვისაც ჭეშმარიტია:

1. ნებისმიერი ორი a, b ტერმისთვის $(a = b) \in P$;
2. თუ $A, B \in P$ პრედიკატებია, ასევე პრედიკატი იქნება $(\neg A), (A \vee B), (A \wedge B), (A \rightarrow B)$;
3. თუ $y \in V$ და $A \in P$, მაშინ $(\exists yA), (\forall yA) \in P$.

მაგალითად $(\forall x0(\forall x1((x0 + x1) = ((x1 + x0) + x10))))$ არითმეტიკული პრედიკატია.

განმარტება 2.10: y იყოს რაიმე ცვლადი.

1. თუ a და b ტერმებია და თუ y ცვლადი ამ რომელიმე ტერმში გვხვდება, მაშინ $(a = b)$ პრედიკატში იგივე y გვხვდება როგორც ***თავისუფალი*** ცვლადი;
2. თუ y ცვლადი რაიმე A არითმეტიკულ პრედიკატში გვხვდება, იგივე ცვლადი $(\exists yA)$ და $(\forall yA)$ პრედიკატებში ***დაბმული*** ცვლადად გვხვდება;
3. თუ A, B არითმეტიკული პრედიკატებია და y ცვლადი მათში თავისუფლად ან დაბმულად გვხვდება, მაშინ იგივე y ცვლადი $(A \vee B), (A \wedge B), (A \rightarrow B)$ და $(\neg A)$ ტერმებში ისევე თავისუფალ ან, შესაბამისად, დაბმულ ცვლადად გვხვდება.

ზედა მაგალითში $x0$ და $x1$ ცვლადები დაბმულად, $x10$ კი როგორც თავისუფალი ცვლადი გვხვდება.
აღსანიშნავია, რომ ჩვენი განსაზღვრებებით დასაშვებია, რომ ერთი და იგივე ცვლადი სხვადასხვა კვანტორში დაბმულ ცვლადად გვხვდებოდეს. იგივე ცვლადი შეიძლება ერთ პრედიკატში სხვადასხვა ადგილზე თავისუფალ ან დაბმულ ცვლადებად გვხვდებოდეს.
შემდგომში $A(x_1, ..., x_n)$ ისეთ პრედიკატს აღნიშნავს, რომელშიც $x_1, ..., x_n$ ცვლადი თავისუფლად გვხვდება.
ისეთ არითმეტიკულ პრედიკატს, რომელშიც თავისუფალი ცვლადი არ გვხვდება, ***არითმეტიკული გამონათქვამი*** ეწოდება.

ლემა 2.2: სიმრავლე $L_E = \{A : A$ არითმეტიკული პრედიკატია$\}$ გამოთვლადია.

სავარჯიშო 2.14: დაწერეთ პროგრამა, რომელიც $w \in \Sigma_E^*$ სიტყვისათვის განსაზღვრავს, არის თუ არა იგი არითმეტიკული პრედიკატი ზემოთ მოყვანილი განმარტების მიხედვით.

აქამდე ჩვენ მხოლოდ სიმბოლოთა მიმდევრობების გარკვეული სიმრავლეები განვსაზღვრეთ. ახლა კი უნდა დავადგინოთ, რა შემთხვევაში იქნება ასეთი მიმდევრობა (სიტყვა) ჭეშმარიტი.

შემდეგი განსაზღვრება ტერმებში ცვლადების კონკრეტული მნიშვნელობების ჩასმის პროცესის ფორმალიზაციის საშუალებას გვაძლევს.

განმარტება 2.11: ცვლადების ჩასმა ეწოდება ფუნქციას $\phi : V \to \mathbb{N}$. თუ რაიმე t ტერმში ϕ ფუნქციის მიხედვით ცვლადების შესაბამის მნიშვნელობებს ჩავსვამთ, ეს აღინიშნება როგორც $\phi(t)$ და რეკურსიულად შემდეგნაირად განისაზღვრება:

$$\phi(0) = 0, \phi(1) = 1, \phi(a + b) = \phi(a) + \phi(b), \phi(a \cdot b) = \phi(a) \cdot \phi(b).$$

მაგალითი: მოცემულია ტერმი $t = ((x0 + x1) \cdot (x0 + x1))$ და $\phi(x0) = 1, \phi(x1) = 2$.
მაშინ $\phi(t) = \phi(x0 + x1) \cdot \phi(x0 + x1) = (\phi(x0) + \phi(x1)) \cdot (\phi(x0) + \phi(x1)) = (1 + 2) \cdot (1 + 2) = 9$.

A პრედიკატის, x ცვლადისა და t ტერმებისთვის შემოვიტანოთ აღნიშვნა $A|_{x=t}$ ისეთი პრედიკატისთვის, რომელიც A პრედიკატში ყოველ თავისუფალ x ცვლადს t ტერმებით შეცვლით.
$A(x_1, ..., x_n)|_{x_1=t_1,...,x_n=t_n}$ ტერმი შემოკლებით ჩავწეროთ როგორც $A(t_1, ..., t_n)$.

$n \in \mathbb{N}$ რიცხვისთვის $\overline{n}$ აღნიშნავს ტერმს $\underbrace{(1 + (... + 1)...)}_{n}$.

განმარტება 2.12: ნებისმიერი a, b ტერმისათვის, A, B პრედიკატისთვის და x ცვლადისთვის ჭეშმარიტია:

- $a = b$ ჭეშმარიტია, თუ $\phi(a) = \phi(b)$ $\forall \phi$ ჩასმისთვის;
- $A \vee B$ ჭეშმარიტია, თუ A ან B (ერთი მაინც) ჭეშმარიტია;
- $A \wedge B$ ჭეშმარიტია, თუ A, B ორივე ჭეშმარიტია;
- $\neg A$ ჭეშმარიტია, თუ A ***არ არის*** ჭეშმარიტი;
- $A \to B$ ჭეშმარიტია, თუ $(B \vee (\neg A))$ ჭეშმარიტია;
- $\exists x A$ ჭეშმარიტია, თუ არსებობს რაიმე $n \in \mathbb{N}$ ისეთი, რომ $A|_{x=\overline{n}}$ ჭეშმარიტია;
- $\forall x A$ ჭეშმარიტია, თუ $A|_{x=\overline{n}}$ ჭეშმარიტია ყველა $n \in \mathbb{N}$-თვის.

მაგალითი: შემდეგი პრედიკატები ჭეშმარიტია:

$((x0 + x1) \cdot (x0 + x1)) = ((x0 \cdot x0) + (((1 + 1) \cdot (x0 \cdot x1)) + (x1 \cdot x1)))$;

$(\exists x1((x0 + x1) = x0))$.

შენიშვნა: სიმარტივისათვის შემდგომ ფორმულებში არ გამოვიყენებთ ფრჩხილების დასმას, ოპერაციათა პრიორიტეტული იერარქია კი შემდეგნაირი იქნება:

$$\cdot, +, =, \forall, \exists, \neg, \wedge, \vee, \to$$

განმარტება 2.13: ლოგიკურ ოპერაციათა აქსიომები და გამოყვანის წესები
A, B და C ირითმეტიკული პრედიკატებისთვის გამოყვანის წესები და აქსიომებია:

1. (ა) $A \to (B \to A)$
1. (ბ) $(A \to B) \to ((A \to (B \to C)) \to (A \to C))$
2. $A\#A \to B\#B$ (დასკვნა)
3. $A \to (B \to A \wedge B)$
4. (ა) $A \wedge B \to A$
4. (ბ) $A \wedge B \to B$
5. (ა) $A \to A \vee B$

5. (ბ) $B \to A \vee B$
6. $(A \to C) \to ((B \to C) \to (A \vee B \to C))$
7. $(A \to B) \to ((A \to \neg B) \to \neg A)$
8. $\neg\neg A \to A$

t ტერმს ეწოდება თავისუფალი y ცვლადის მიმართ $A(y)$ პრედიკატში, თუ $A(t)$-ში t ტერმში შემავალი ყველა ცვლადი თავისუფალია (სხვა სიტყვებით რომ ვთქვათ, A პრედიკატში t ტერმის ჩასმის შემდეგ ისეთ პრედიკატს ვიღებთ, რომელშიც t ტერმის ყველა ცვლადი თავისუფალია).

მაგალითი:

$$A(x1) = (\exists x0(1 + 1 + x0 = x1)) \to (\exists x0(1 + x0 = x1))$$

პრედიკატში ტერმები $1, 1 + 1$ და $x10$ თავისუფალია $x1$ ცვლადის მიმართ, ხოლო $x0, 1 + 1 + x0$ კი არა.

შენიშვნა: თუ t არ არის თავისუფალი y ცვლადის მიმართ $A(y)$ პრედიკატში, მაშინ $A(t)$ შეიძლება მცდარი გახდეს იმ შემთხვევაშიც კი, თუ $A(y)$ ჭეშმარიტია. ზემოთ მოყვანილი $A(y)$ ჭეშმარიტია, მაგრამ $A(1+1+x0)$ კი მცდარია.

განმარტება 2.14: აქსიომები და გამოთვლის წესები კვანტორებისათვის
ნებისმიერი $A(y)$ (რომელშიც y თავისუფალია) პრედიკატის, t (რომელიც y ცვლადის მიმართ თავისუფალია $A(y)$-ში) ტერმისა და C პრედიკატისთვის (რომელშიც y არაა თავისუფალი) გამოთვლის წესი და აქსიომაა:

9. $C \to A(y) \# C \to \forall y A(y)$
10. $\forall y A(y) \to A(t)$
11. $A(t) \to \exists y A(y)$
12. $A(y) \to C \# \exists y A(y) \to C$

განმარტება 2.15: ნატურალურ რიცხვთა აქსიომები
ნებისმიერი $A(y)$ (რომელშიც y თავისუფალია) პრედიკატისა და a, b ტერმისთვის აქსიომებია

13. $A(0) \wedge \forall y(A(y) \to A(y+1)) \to A(y)$ — ინდუქციის აქსიომა
14. $a + 1 = b + 1 \to a = b$
15. $\neg(a + 1 = 0)$
16. $a = b \to (a = c \to b = c)$
17. $a = b \to a + 1 = b + 1$
18. $a + 0 = a$
19. $a + (b + 1) = (a + b) + 1$

} მიმატების აქსიომები

20. $a \cdot 0$
21. $a \cdot (b + 1) = a \cdot b + a$

} გამრავლების აქსიომები

A_E განვსაზღვროთ, როგორც $1, 3 - 8, 10 - 11, 13 - 21$ აქსიომების ჩანაწერები, ხოლო S_E კი როგორც $2, 9$ და 12 წესების ჩანაწერები.

ადვილად შეიძლება პროგრამების დაწერა, რომლითაც ნებისმიერი w სიტყვისთვის დავადგენთ, არის თუ არა იგი A_E ან S_E სიმრავლეში.
ლემა 1.2-დან გამომდინარე (რომლის მიხედვითაც Z_E გამოთვლადია), შესაძლებელია ზემოთ ჩამოყალიბებული პრინციპების მქონე $Z_E = (\Sigma_E, L_E, A_E, S_E)$ გამოთვლადი დამტკიცების სისტემის აგება.

მაგალითი: განვიხილოთ $x0 = x0$ პრედიკატის დამტკიცების ჯაჭვი Z_E სისტემაში.

1. $x0 = x1 \to (x0 = x10 \to x1 = x10)\#$ (აქსიომა 16)
2. $0 = 0 \to (0 = 0 \to 0 = 0)\#$ (აქსიომა 1 (ა))

3. $(x0 = x1 \to (x0 = x10 \to x1 = x10)) \to ((0 = 0 \to (0 = 0 \to 0 = 0)) \to (x0 = x1 \to (x0 = x10 \to x1 = x10)))\#$ (აქსიომა 1 (ა))
4. $(0 = 0 \to (0 = 0 \to 0 = 0)) \to (x0 = x1 \to (x0 = x10 \to x1 = x10))\#$ (წესი 2)
5. $(0 = 0 \to (0 = 0 \to 0 = 0)) \to \forall x10(x0 = x1 \to (x0 = x10 \to x1 = x10))\#$ (აქსიომა 9)
6. $(0 = 0 \to (0 = 0 \to 0 = 0)) \to \forall x1\forall x10(x0 = x1 \to (x0 = x10 \to x1 = x10))\#$ (აქსიომა 9)
7. $(0 = 0 \to (0 = 0 \to 0 = 0)) \to \forall x0\forall x1\forall x10(x0 = x1 \to (x0 = x10 \to x1 = x10))\#$ (აქსიომა 9)
8. $\forall x0\forall x1\forall x10(x0 = x1 \to (x0 = x10 \to x1 = x10))\#$ (წესი 2)
9. $\forall x0\forall x1\forall x10(x0 = x1 \to (x0 = x10 \to x1 = x10)) \to \forall x1\forall x10(x0 + 0 = x1 \to (x0 + 0 = x10 \to x1 = x10))\#$ (აქსიომა 10)
10. $\forall x1\forall x10(x0 + 0 = x1 \to (x0 + 0 = x10 \to x1 = x10))\#$ (წესი 2)
11. $\forall x1\forall x10(x0 + 0 = x1 \to (x0 + 0 = x10 \to x1 = x10)) \to \forall x10(x0 + 0 = x0 \to (x0 + 0 = x10 \to x0 = x10))\#$ (აქსიომა 10)
12. $\forall x10(x0 + 0 = x0 \to (x0 + 0 = x10 \to x0 = x10))\#$ (წესი 2)
13. $\forall x10(x0 + 0 = x0 \to (x0 + 0 = x10 \to x0 = x10)) \to (x0 + 0 = x0 \to (x0 + 0 = x0 \to x0 = x0))\#$ (აქსიომა 10)
14. $x0 + 0 = x0 \to (x0 + 0 = x0 \to x0 = x0)\#$ (წესი 2)
15. $x0 + 0 = x0\#$ (აქსიომა 18)
16. $x0 + 0 = x0 \to x0 = x0\#$ (წესი 2)
17. $x0 = x0\#$ (წესი 2)

ამ მაგალითიდან ჩანს, რომ თვით უმარტივესი ტერმებისთვისაც კი დამტკიცების ჯაჭვი საკმაოდ გრძელი შეიძლება იყოს, მაგრამ ამ ფორმალიზმში მარტივი წესების მეშვეობით საკმაოდ რთული თეორემების დამტკიცება შეიძლება (მაგალითად შესაძლებელია ელემენტარული რიცხვთა თეორიის ლექციებში მოყვანილი ყველა თეორემის დამტკიცება). ტიურინგის მანქანების მსგავსად აქაც ფორმალიზმი საკმაოდ რთულად შეიძლება მოგვეჩვენოს, მაგრამ გარკვეული ვარჯიშის შემდეგ ინტუიცია ვითარდება, რითაც ფორმალური ჯაჭვების აგების გარეშე შეიძლება მივხვდეთ, დამტკიცებადია თუ არა რაიმე გამონათქვამი (ასევე გარკვეული შეჩვევის შემდეგ ტიურინგის მანქანების კონკრეტულად აგების გარეშე შეგვიძლია მივხვდეთ, გამოთვლადია თუ არა რაიმე ამოცანა).
აქვე უნდა აღინიშნოს, რომ ჩვენ კონკრეტული ტერმის დამტკიცება მოვიყვანეთ და გადავამოწმეთ. სხვა საკითხია, თუ ოგორ შეიძლება რაიმე ტერმის დამტკიცების პოვნა, რაც ზოგად შემთხვევაში არაამოხსნადი და ხშირ კონკრეტულ შემთხვევაში საკმაოდ რთულია.

2.3 ჭეშმარიტება, არაწინააღმდეგობრივობა და სისრულე

განმარტება 2.16: ნებისმიერი დამტკიცების სისტემა $\mathcal{W} = (\Sigma, L, A, S)$ არის ***ჭეშმარიტი***, თუ ყველა მასში გამოყვანადი გამონათქვამი ჭეშმარიტია; ***არაწინააღმდეგობრივი***, თუ არც ერთი $w \in L$ სიტყვისთვის როგორც w, ასევე მისი უარყოფის $\neg w$ გამოყვანა არ შეიძლება; ***სრული***, თუ ნებისმიერი $w \in L$ სიტყვისთვის ან w ან მისი უარყოფა $\neg w$ გამოყვანადია.
ნებისმიერი ჭეშმარიტი სისტემა ამავდროულად არაწინააღმდეგობრივი უნდა იყოს. ჭეშმარიტებისგან განსხვავებით არაწინააღმდეგობრივობა არ განიხილავს ტერმთა ჭეშმარიტების ან მნიშვნელობის საკითხებს, არამედ დამტკიცების სისტემების წმინდა კომბინატორული პრობლემაა. აქედან გამომდინარე, ეს საკითხი ფორმალური სისტემებისთვის უფრო ადვილი გადასაჭრელია, ვიდრე ჭეშმარიტება.
ინტუიციურად რაიმე W სისტემის კორექტულობა შემდეგნაირად შეიძლება დამტკიცდეს:

- ვაჩვენოთ, რომ $\mathcal{W}$ სისტემის აქსიომები ჭეშმარიტია;
- ვაჩვენოთ, რომ $\mathcal{W}$ სისტემის გამოთვლის წესები ჭეშმარიტია;

- ზედა ორი დებულებიდან ინდუქციით გამომდინარეობს, რომ $\mathcal{W}$ სისტემის ყველა გამოყვანილი ტერმი ჭეშმარიტია.

აღსანიშნავია, რომ აქსიომათა და გამოყვანის წესების ჭეშმარიტების ინტუიციურად დაჯერება შეიძლება არასწორიც აღმოჩნდეს: რამოდენიმე ცნობილი სისტემა, რომლებიც დიდწილად ჭეშმარიტად იყო მიჩნეული, საბოლოოდ მცდარი აღმოჩნდა (იხ. რასელის პარადოქსი).
ზემოთ მოყვანილი Z_E სისტემა ბევრი (მაგრამ არა ყველა) მეცნიერის მიერ ჭეშმარიტადაა მიჩნეული (დავის საგანია აქსიომა $\neg\neg A \to A$).
ნებისმიერ ჭეშმარიტ და სრულყოფილ სისტემაში ნებისმიერი ამ სისტემის ენაზე ფორმულირებული ტერმის ჭეშმარიტების დადგენა ალგორითმულად შეიძლება:

ლემა 2.3: ნებისმიერ ჭეშმარიტ და სრულ $\mathcal{W} = (\Sigma, L, A, S)$ დამტკიცების სისტემაში სიმრავლე $\{w \in L : w$ ჭეშმარიტია$\}$ გამოთვლადია.

დამტკიცება: დასამტკიცებლად მოვიყვანოთ ალგორითმი, რომელიც ამ სიმრავლეს გამოითვლის. პირველ რიგში უნდა აღინიშნოს, რომ (ყოველივე ზემოთ თქმულიდან გამომდინარე) ნებისმიერი $b \in (\Sigma \cup \{\#\})^*$ ტერმისთვის შესაძლებელია იმის გადამოწმება, არის თუ არა იგი რაიმე ტერმის ჭეშმარიტების დამტკიცება. აქედან გამომდინარე შეიძლება $(\Sigma \cup \{\#\})^*$ სიმრავლის დალაგება და ნებისმიერი $w \in L$ მონაცემისთვის იმის დადგენა, მისი დამტკიცება არსებობს სისტემაში თუ მისი $\neg w$ უარყოფის (ამისათვის საჭიროა მხოლოდ დალაგებული სიმრავლის სიტყვების რიგ-რიგობით გადამოწმება). რადგან თავდაპირველი დამტკიცების სისტემა სრულია, ნებისმიერი $w \in L$ გამონათქვამი მასში ან ჭეშმარიტია, ან მცდარი და, აქედან გამომდინარე, იარსებებს ან მისი, ან მისი უარყოფის დამტკიცება და ჩვენი ალგორითმიც ყოველთვის გაჩერდება. რადგან პირობის თანახმად $\mathcal{W}$ სისტემა ჭეშმარიტია, ჩვენი ალგორითმიც სწორ პასუხს დააბრუნებს.

რ.დ.გ.

2.4 გიოდელის არასისრულის თეორემები

თეორემა 2.5: $\mathcal{W} = (\Sigma, L, A, S)$ იყოს რაიმე დამტკიცების სისტემა და დავუშვათ, რომ არსებობს გამოთვლადი ფუნქცია $H : \{0, 1, ||\}^* \to L$ ისეთი, რომ $\forall u, v, \in \{0, 1\}^*$ სიტყვისთვის ფუნქცია $H(u||v)$ მაშინ და მხოლოდ მაშინ იძლევა $\mathcal{W}$ სისტემაში ჭეშმარიტ ტერმს, როდესაც ერთკვალიანი ტიურინგის მანქანა M_u გაჩერდება v მონაცემზე. მაშინ $\{h \in L : h$ ჭეშმარიტია$\}$ სიმრავლე არაა გამოთვლადი.

დამტკიცება: ადვილი საჩვენებელია, რომ გაჩერების ამოცანა დაიყვანება $\{h \in L : h$ ჭეშმარიტია$\}$ სიმრავლეზე.

სავარჯიშო 2.15: დაიყვანეთ გაჩერების ამოცანა $\{h \in L : h$ ჭეშმარიტია$\}$ სიმრავლეზე.

შენიშვნა: ზემოთ მოყვანილი თეორემის პირობა გვეუბნება, რომ შესაძლებელია $H(u||v) \in L$ გამოთვლა, მაგრამ ზოგად შემთხვევაში შეუძლებელია იმის დადგენა, ჭეშმარიტია თუ არა ეს სიტყვა L ენაში (ამით თეორემა არ ეწინააღმდეგება გაჩერების ამოცანის გამოთვლის შეუძლებლობას).

დასკვნა: ნებისმიერი ჭეშმარიტი დამტკიცების სისტემა $\mathcal{W}$, რომელიც წინა თეორემის პირობებს აკმაყოფილებს, არაა სრული (ეს წინა თეორემისა და ლემის შედეგად გამომდინარეობს).

ლემა 2.4: (ძირითადი ლემა) Z_E ასრულებს წინა თეორემის პირობებს.

დამტკიცება: შემდეგში იქნება მოყვანილი

ამით მტკიცდება

თეორემა 2.6: ჭეშმარიტ არითმეტიკულ გამონათქვამთა სიმრავლე ***არაა გამოთვლადი.*** თუ Z_E ჭეშმარიტია, მაშინ მასში იარსებებს ისეთი გამონათქვამი, რომელსაც ვერც დავამტკიცებთ და ვერც უარვყოფთ.

შეიძლება იმის იმედი გვქონდეს, რომ არასრულ სისტემას (მაგალითად Z_E) ისეთი აქსიომებით გავაფართოვებთ, რომ აქამდე არაამოხსნადი ამოცანების გადაჭრა შევძლოთ (ცხადია, რომ ასეთი დამატებითი აქსიომები გამოთვლადი უნდა იყოს).

იტყვიან, რომ რაიმე სისტემა $\mathcal{W} = (\Sigma, L, A, S)$ მოიცავს Z_E სისტემას, თუ $L_E \subset L$, $A_E \subset A$ და $S_E \subset S$. რადგან ზემოთ მოყვანილი თეორემები და მათი შედეგები ყველა სისტემისთვისაა ჭეშმარიტი, რომელიც Z_E სისტემას მოიცავს, ჭეშმარიტია შემდეგი

თეორემა 2.7: ნებისმიერი ჭეშმარიტი სისტემა, რომელიც Z_E სისტემას მოიცავს, ***არაა სრულლი.***

სხვა სიტყვებით რომ ვთქვათ, ნებისმიერი არასრული სისტემის „შევსება" შეიძლება აქსიომებით, რომელთა საშუალებით აქამდე ამოუხსნად საკითხებს გადავჭრიდით, მაგრამ სამაგიეროდ ახალი არაამოხსნადი გამონათქვამები გაგვიჩნდება.

ზემოთ მოყვანილი ყველა შედეგი არსებობის თეორემაა - ჩვენ ვიცით ამოუხსნადი ამოცანების არსებობის შესახებ, მაგრამ კონკრეტულად მათი პოვნა შეუძლებელია. ერთ-ერთი ამოუხსნადი ამოცანის აღწერა შესაძლებელია ***გიოდელის არასისრულის*** თეორემის მეშვეობით:

თეორემა 2.8: ნებისმიერ ჭეშმარიტ $\mathcal{W} = (\Sigma, L, A, S)$ დამტკიცების სისტემაში, რომელიც 1.1 თეორემის პირობებს აკმაყოფილებს, შეიძლება კონკრეტული გამონათქვამების მოყვანა, რომელთა (ამ სისტემაში) არც დამტკიცება და არც უარყოფაა შესაძლებელი. ამას გარდა, თუ $\mathcal{W}$ მოიცავს Z_E სისტემას, გამონათქვამის მცდარობა შეიძლება ინტუიციურად, ანუ $\mathcal{W}$ სისტემის ფორმალიზმის გარეშე ვაჩვენოთ.

დამტკიცება: Q იყოს ტიურინგის მანქანა, რომელიც $w \in \{0,1\}^*$ მონაცემით $\mathcal{W}$ სისტემის ყველა დამტკიცებას გადასინჯავს და გაჩერდება მაშინ და მხოლოდ მაშინ, თუ იპოვნის დამტკიცებას $\neg H(w||w)$ ტერმისთვის. თუ q ამ Q მანქანის გიოდელის რიცხვია, $H(q||q)$ არ უნდა იყოს გამოთვლადი.
მართლაც, თუ დავუშვებთ, რომ $H(q||q)$ $\mathcal{W}$ სისტემაში დამტკიცებადია, $\mathcal{W}$ სისტემის ჭეშმარიტების და, აქედან გამომდინარე, არაწინააღმდეგობრივობის გამო, $\neg H(q||q)$ დამტკიცება ამ სისტემაში ვერ იარსებებდა. მაგრამ ამ შემთხვევაში კი Q მანქანა q მონაცემით არ გაჩერდებოდა (თავისი კონსტრუქციის გამო - იგი $\neg H(q||q)$ მკიცებას ვერ იპოვნიდა). აქედან გამომდინარე, $\mathcal{W}$ სისტემა არ იქნებოდა ჭეშმარიტი, რადგან მცდარი გამონათქვამი $H(q||q)$ დამტკიცებადი გამოვიდოდა.
მეორეს მხრივ, თუ დავუშვებთ, რომ $\neg H(q||q)$ დამტკიცებადია $\mathcal{W}$ სისტემაში, Q მანქანა თავისი კონსტრუქციის თანახმად q მონაცემზე გაჩერდებოდა და ისევ ანალოგიურ წინააღმდეგობას მივიღებთ.

შენიშვნა: გავიხსენოთ, რომ $H(q||q)$ ტოლფასია გამონათქვამისა „Q ტიურინგის მანქანა გაჩერდება q მონაცემზე" და, შესაბამისად, $\not{H}(q||q)$ ნიშნავს „Q ტიურინგის მანქანა არ გაჩერდება q მონაცემზე".

თეორემის მეორე ნაწილის დასამტკიცებლად გამოვიყენოთ

ლემა 2.5: თუ $H(u||v)$ ჭეშმარიტ გამონათქვამს იძლევა, მაშინ ეს გამონათქვამი დამტკიცდება Z_E სისტემაში.

დამტკიცება: ინტუიციურად M_u მანქანის მიერ v მონაცემით სასრულ ბიჯებში პასუხის გამოთვლა იმის დასტურია, რომ M_u გაჩერდება v მონაცემზე. ასეთი გამოთვლიდან შესაძლებელია Z_E სისტემაში $H(u||v)$ სისწორის ფორმალური დამტკიცების აგება (ამის ფორმალურად დამტკიცება საკმაოდ გრძელ გამოთვლას მოითხოვს).
რ.დ.გ.

ამ ლემიდან პირდაპირ გამომდინარეობს გიოდელის თეორემის მეორე ნაწილი, რადგან Q მანქანა q მონაცემზე რომ გაჩერებულიყო, მაშინ ასევე $H(q||q)$ იქნებოდა Z_E სისტემაში გამოყვანადი, რაც ამავე თეორემის პირველი ნაწილის დამტკიცებით შეუძლებელია.
რ.დ.გ.

ახლა კი განვიხილოთ არაწინააღმდეგობრივობის დამტკიცების შესაძლებლობის საკითხი. რადგან არაწინააღმდეგობრივობას აქსიომებისა და გამოყვანის წესების შინაარსთან კავშირი არ აქვს, შეიძლება იმის იმედი გვქონდეს, რომ შედარებით მარტივი სისტემებით (მაგ. Z_E) უფრო დიდი სისტემების (მაგ. სიმრავლეთა თეორიის) არაწინააღმდეგობრივობა დავამტკიცოთ.
ეს ამოცანა ასიოდე წლის წინ დაისვა და „ჰილბერტის პროგრამის" სახელითაა ცნობილი. ასე პირდაპირ ისიც არაა ნათელი, შეიძლება თუ არა რაიმე $\mathcal{W}$ სისტემის ფარგლებში გამონათქვამის „$\mathcal{W}$ არაწინააღმდეგობრივია" ფორმალურად ჩამოყალიბება. მაგრამ ადვილად შეიძლება იმის ჩვენება, რომ ამ გამონათქვამის ფორმალური აღწერა უკვე Z_E სისტემაშიც კი არის შესაძლებელი.

ლემა 2.6: ნებისმიერი $\mathcal{W}$ დამტკიცების სისტემისთვის არსებობს არითმეტიკული ტერმი, რომლის მნიშვნელობა იქნება „$\mathcal{W}$ სისტემა არაწინააღმდეგობრივია".

დამტკიცება: R ყიოს ისეთი ტიურინგის მანქანა, რომელიც მონაცემით 0 მოცემული $\mathcal{W}$ სისტემის ყველა დამტკიცებას რიგ-რიგობით გაივლის, ახალ დამტკიცებას მეხსიერების კვალზე ჩაწერს და ყოველ ახალ დამტკიცებას w აქამდე დამტკიცებულებს შეადარებს. ეს მანქანა გაჩერდება მაშინ და მხოლოდ მაშინ, თუ აღმოაჩენს ისეთ w და w' გამონათქვამთა წყვილს, რომ $w = \neg w'$ ან $w' = \neg w$.
თუ r რიცხვი R მანქანის გიოდელიზაციაა, მაშინ გამონათქვამი „$\mathcal{W}$ არაწინააღმდეგობრივია" ფორმალურად გამოისახება, როგორც $\neg H(r||0)$.

რ.დ.გ.

შენიშვნა: სიმარტივისათვის „$\neg H(r||0)$" ჩავწეროთ როგორც „Consis $\mathcal{W}$".

ახლა კი დავამტკიცოთ ***გიოდელის არასისრულის მეორე თეორემა.***

თეორემა 2.9: ნებისმიერ არაწინააღმდეგობრივ დამტკიცების სისტემაში $\mathcal{W} = (\Sigma, L, A, S)$, რომელიც Z_E სისტემას მოიცავს, „Consis $\mathcal{W}$" დამტკიცება შეუძლებელია.

დამტკიცება: პირველი თეორემის ანალოგიურად Q იყოს ისეთი ტიურინგის მანქანა, რომელიც w მონაცემზე ყველა დამტკიცებას გადაითვლის და გაჩერდება მაშინ და მხოლოდ მაშინ, თუ $\neg H(w||w)$ დამტკიცებას იპოვნის. q იყოს მისი გიოდელიზაცია.

ლემა 2.7: თუ $\mathcal{W}$ არაწინააღმდეგობრივია, მაშინ $\neg H(q||q)$ არაა გამოყვანადი $\mathcal{W}$ სისტემაში.

დამტკიცება: $\neg H(q||q)$ რომ ყოფილიყო გამოყვანადი $\mathcal{W}$ სისტემაში, მაშინ Q მანქანა q მონაცემზე იპოვნიდა $\neg H(q||q)$ დამტკიცებას და გაჩერდებოდა. მაგრამ ამ შემთხვევაში 1.5 ლემის თანახმად აგრეთვე $H(q||q)$ იქნებოდა Z_E და, აქედან გამომდინარე, $\mathcal{W}$ სისტემაში გამოყვანადი და $\mathcal{W}$ წინააღმდეგობრივი გამოვიდოდა.

რ.დ.გ.

ასევე ჭეშმარიტია

ლემა 2.8: ნებისმიერი $v \in L$ სიტყვისათვის არსებობს არითმეტიკული პრედიკატი, რომელიც აღწერს წინადადებას „v გამოყვანადია $\mathcal{W}$ სისტემაში".

დამტკიცება: M იყოს ისეთი ტიურინგის მანქანა, რომელიც v მონაცემზე $\mathcal{W}$ სისტემის ყველა დამტკიცებას გადაარჩევს და გაჩერდება მაშინ და მხოლოდ მაშინ, თუ აღმოაჩენს v გამონათქვამის დამტკიცებას. თუ m ამ მანქანის გიოდელიზაციაა, მაშინ $H(m||v)$ იქნება საძიებო წინადადების არითმეტიკული პრედიკატი.

რ.დ.გ.

ჭეშმარიტია შემდეგი

ლემა 2.9:

1. გამონათქვამი „Consis $\mathcal{W} \to (\neg H(q||q)$ არაა გამოყვანადი $\mathcal{W}$ სისტემაში)" გამოყვანადია Z_E (და, აქედან გამომდინარე, $\mathcal{W}$) სისტემაში.

დამტკიცება: წინა ლემების დამტკიცებების ფორმალიზაცია Z_E სისტემაში.

რ.დ.გ.

„Consis $\mathcal{W}$" გამოყვანადი რომ ყოფილიყო $\mathcal{W}$ სისტემაში, წინა ლემიდან დავასკვნიდით, რომ

2. „$\neg H(q||q)$ არ არის გამოყვანადი $\mathcal{W}$ სისტემაში"

და Q მანქანის განსაზღვრების გათვალისწინებით,

3. „$\neg H(q||q)$"

რაც ასევე Z_E სისტემაში იქნება ჭეშმარიტი.

ლემა 2.10: თუ $v \in L$ გამოყვანადია $\mathcal{W}$ სისტემაში, მაშინ ასევე Z_E სისტემაში გამოყვანადი იქნება არითმეტიკული გამონათქვამი მნიშვნელობით „v გამოყვანადია $\mathcal{W}$ სისტემაში".

დამტკიცება: უშუალოდ გამომდინარეობს წინა ლემებიდან.

რადგან „$\neg H(q||q)$" გამოყვანადი იქნებოდა $\mathcal{W}$ სისტემაში, Z_E სისტემაში ასევე გამოყვანადი იქნებოდა გამონათქვამი

4. „$\neg H(q||q)$ გამოყვანადია $\mathcal{W}$ სისტემაში".

ასე რომ, (2) და (4) გამოყვანადია $\mathcal{W}$ სისტემაში. რადგან (2) = $\neg$(4), $\mathcal{W}$ სისტემის არაწინააღმდეგობრივობის დამტკიცება შეუძლებელია.

რ.დ.გ.

გიოდელის მიერ დამტკიცებული არასისრულის თეორემების შემდეგ რადიკალურად შეიცვალა ათასწლეულების მანძილზე არსებული შეხედულება მეცნიერების მიმართ. თუ მანამდე მიჩნეული იყო, რომ ისეთ ზუსტ მეცნიერებაში, როგორიც მათემატიკაა, შესაძლებელია ნებისმიერი რამის ან დამტკიცება, ან უარყოფა, აღმოჩნდა, რომ თვით არითმეტიკაში - ადამიანების მიერ შექმნილ წარმოსახვით სამყაროშიც კი, სადაც ადამიანების მიერვე შემოტანილი წესები მოქმედებს, ყველაფრის გაგება შეუძლებელია.
საინტერესო ფაქტია, რომ თუ არითმეტიკის აქსიომათა სისტემაში უარს ვიტყვით გამრავლების აქსიომებზე, მივიღებთ ე.წ. ***„პრესბურგერის არითმეტიკას"***, რომელიც მისი ავტორის - მოშე პრესბურგერის სახელს ატარებს. როგორც დამტკიცებულია, ეს არითმეტიკა სრულყოფილია: მასში ყველა გამონათქვამის ან დამტკიცება, ან უარყოფაა შესაძლებელი, მაგრამ ამ დამტკიცების პროცედურა ძალიან გრძელია. n სიმბოლოიანი გამონათქვამის დამტკიცების ქვედა ზღვარი ორმაგად ექსპონენციურია: $\Omega(2^{2^{cn}})$, $c \in \mathbb{N}$.
ფილოსოფიური თვალსაზრისით ეს ნიშნავს, რომ ადამიანი ყველაფერს ვერ ჩაწვდება. იმის გათვალისწინებით, რომ არაგამოთვლად ამოცანათა სიმრავლე არაა თვლადი, შეიძლება დავასკვნათ, რომ ადამიანი მოვლენათა უმეტესობას ვერ ჩაწვდება.

3 დასკვნა

Man muß die Welt nicht verstehen,
man muß sich nur darin zurechtfinden.
Albert Einstein

სამყაროს შემეცნება არ არის აუცილებელი;
საკმარისია მასში ორიენტაცია შევძლოთ.
ალბერტ აინშტაინი

სირთულის თეორიის გამოყენებით შეიძლება დავრწმუნდეთ, რომ ადამიანი მის გარშემო არსებულ სამყაროს ვერ ჩაწვდება. უფრო მეტიც: მის მიერვე გამოგონებულ სისტემებშიც კი (მაგალითად, მათემატიკაში და, აქედან გამომდინარე, დანარჩენ მეცნიერებებში) მნიშვნელოვან საკითხებს ვერ გაარკვევს. მაგალითად, ფუნდამენტურ შეკითხვებზე პასუხის გაცემა შეუძლებელია. უფრო მეტიც: ბევრი ყოველდღიური ამოცანა არსებობს ჩვენს გარშემო, რომლის გადაჭრაც შეუძლებელი ან იმდენად რთულია, რომ პრაქტიკულად შეუძლებელი ხდება. ის ფაქტიც კი არაა ცნობილი, შეძლებს თუ არა ვინმე მათ რეალურ დროში გადაჭრას. მეორეს მხრივ, იგივე სირთულის თეორია გვეუბნება, თუ როგორ შეიძლება ამ ფუნდამენტური პრობლემების გადაჭრა: მიახლოებითი ამოხსნის ძიება, შემთხვევითი (სტოქასტური) ალგორითმების შედგენა, შესაბამისი ევრისტიკების აგება და ა. შ.
გარდა ასეთი პრაქტიკული შედეგებისა, სირთულის თეორიაზე დაყრდნობით სხვადასხვა მათემატიკური (ლოგიკური) თეორემის მტკიცება გაცილებით უფრო ადვილი და გასაგები ხდება (მაგალითად ზემოთ გარჩეული გიოდელის თეორემები არითმეტიკის არასრულყოფილებისა და არაწინააღმდეგობრივობის დამტკიცების შეუძლებლობის შესახებ).
სირთულის თეორიაზე დაყრდნობით ასევე დამტკიცდა, რომ ისეთი ამოცანებიც არსებობს, რომელთა ამოხსნის გზა ჩვენს გარშემო მყოფ ამოცანათა უმეტესობის პასუხს მოგვცემდა. ასეთ ამოცანებს NP-სრული ეწოდება და მათი ამოხსნის გზის ცოდნა ჩვენს ცოდნას „გაანათებდა". ეს რაღაც თვალსაზრისით ზღაპრებში არსებული სიბრძნის ქვის ტოლფასია, რაც რეალობაშიც შესაძლებელი ყოფილა.
მოკლედ რომ ვთქვათ, სირთულის თეორიის მეშვეობით მტკიცდება, რომ ჩვენ ჩვენს სამყაროს ვერ ჩავწვდებით, მაგრამ ამავე თეორიის გამოყენებით მასში ორიენტაცია უფრო ადვილადაა შესაძლებელი.