İÇİNDEKİLER

1. ROBOTİK TARİHİ

Robot, kontrol ve algılama sistemleriyle bilgisayar algoritmalarına bağlı olarak akıllı davranan makinedir. Amerikan Robot Enstitüsü tarafından ise robot şu şekilde tanımlanmaktadır: "Robot, yeniden programlanabilen; maddeleri, parçaları, aletleri, özel cihazları yerinden oynatabilen çok fonksiyonlu makinedir".

Medeniyet, kölelik düzeni sayesinde bugünkü olağanüstü seviyeye ulaştı. Bir köle tükettiğinden fazlasını üretmek zorundadır. Bunun sonucu olarak bir artı değer elde edilir ve bu artı değer de insanlığın gelişmesi için itici güç sağlar. Kölelik sayesinde kalıcı anıtlar, dev yapılar, şehirler yapıldı ve yine bu sayede bilginler geçim derdi düşünmeden kendilerini işlerine verebildiler çünkü köleler onlar için fazlasıyla üretiyorlardı. Robot hayalinin ekonomik nedeni işte bu "köle" elde etme ihtiyacıdır.

Eskiçağ Mısır'ı rahipleri dev tanrı heykellerinin yapımını tanrı iradesiyle hareket ettikleri gerekçesiyle yöneterek insanların sadakatle hizmet etmesini sağlamışlardır. Bunun gibi birçok olay insanlar üzerinde büyük etki bıraktı. Her dediğini yapan hizmetçilere sahip olmanın insan egosunu tatmin ettiği bir gerçektir. Bu da robot sahibi olma isteğinin psikolojik nedenidir.

Son olarak günümüzün insan haklarındaki düzenlemeleri insanın birey olarak önemini arttırdı; çalışma şartlarının iyileştirilmesi, ücret, emeklilik ve kaza halinde tazminat gibi faktörler önem kazandı.

İnsanların istenmeyen ve tehlikeli olabilecek işlerde çalışmaması gerektiği gündeme geldi. Peki o zaman bu işleri kim yapacaktı? Bu da robot özleminin sosyal nedenidir.

Yunan uygarlığı zamanında İskenderiye'de hidrolik düzenlere sahip hareketli kuklalar vardı. Bunlar, insana benzetmek amacıyla yapılmayıp hidrolik bilimi eğitiminde deney aracıydı ve tapınaklarda saklanırdı. Arada sırada da tapınağa gelenleri etkilemek veya eğlendirmek için kullanılırdı.

Ortaçağda Selçuklu Türklerinin Sükman boyundan Cizreli Ebul-İz, yalnız suyun potansiyel ve kinetik enerjilerinden faydalanarak makine ve robotlar yaptı. İlkel otomatlar 17. ve 18. yüzyıllarda Avrupa'da bulundu. Bunlar birer mekanik harikasıydılar. Zamanımızdan 350 yıl evvelinden mucitler ilginç makineler icat edip üretime geçtiler. Kilise ve katedrallerin tepesinde bulunan devasa saatlerde gerçek boyutlarında insan, melek, şeytan gibi figürler vardı. Bunlar ellerindeki çana bir tokmakla vurarak saati belirtirlerdi. Williams ve Penniman bu gibi gelişmeler hakkında detaylı bilgi verdiler.

Bazıları şöyledir:

1) **Ötücü kuş:** Kurulu düzenek tarafından miller ve kaldıraçlar yardımıyla kuşun kanatları, kafası ve gagası kontrol ediliyordu. Ayrıca vana ve pistonlar sayesinde kuş sesi çıkarıyordu. Çalışırken kafasını ve kanatlarını hareket ettirip öterken de gagasını oynatıyordu. Bundan çok sayıda üretildi ve ev dekorasyonunda kullanıldı. Kuşun hareketlerini sağlayan,miller (ki bunlar bir çeşit salt okunur bellek -ROM- gibidir) ve mekanik sürücüdür. Elektronik karşılığı ROM birimi ve küçük bir elektrik motorunu kontrol eden transistorlu sürücülerdir.

2) **Otomatik Flütçü:** Müzisyen kıyafeti giyen otomatik flütçü, dudaklarına yapışık flüte hava üflerken parmaklarıyla da flütün deliklerini açıp kapatırdı. Millerle sağlanan bir takım hareketler müzik kutusu tamburuna benzer, dilimlenmiş karmaşık bir tamburla yönlendiriliyordu. Bu flütçü 1738 yılında Jacques de Vaucanson tarafından Paris'de yapılmıştı. O zamanlarda metal işlemesi yapan makineler olmadığından tamamı el işiydi. Vaucanson yaptığı bu otomatı bir binada sergiledi ve insanlar bunu görebilmek için çok uzaklardan bile geldiler. Hatta krallar ve imparatorlar bile özellikle memnuniyetlerini belirttiler. Bu yüzden pek çok kabiliyetli mucit otomatlar üzerine çalışmaya başladı.

1.

3) **Maillardet'in Otomatı:** Belki de kurulu düzenekli otomatlar içinde en karmaşık olanı 1805 yılında Londra'da Henri Meillardet tarafından yapılan yazı yazabilen ve resim yapabilen inanılmaz otomattır. Geniş belleği ve titiz hareketleri vardı. Örneğin bir gemi resmini aslına uygun, bütün detaylarıyla beş dakikada çizebiliyordu. Repertuarında pekçok örnek vardı ve bunlardan biri de Fransız stilinde yazılmış beş dizelik şiirdi. Bu otomat halen Franklin Enstitüsünde sergilenmektedir.

Buraya kadar anlatılan otomatların genel özellikleri şunlardır:

1) Eğlence amaçlı yapılmışlardı. İnsanlar hayran olmaya ve gördüklerini arkadaşlarına anlatmak için gelirlerdi. Toplumun ilgisi, çalışma metodunu ve bunun manasını anlamaktan çok gösteri amaçlıydı.
2) Geribesleme ve duyarga yoktu. Karmaşık hareketler ardışıl olarak yapılırdı fakat çevreye bir tepki sözkonusu değildi.

Bir makine, iç ve dış bozucu mevcutsa, belirli bir çıkış üretebilmesi için kontrol sistemine ihtiyaç duyar. Bu kontrol sisteminin çalışabilmesi için de iç ve dış algılayıcılar gereklidir. Örnek olarak üstte tanıtılan yazı yazma otomatı ele alınabilir. Eğer algılayıcı geribeslemesi yoksa, "yaz" komutu verilirse ve de kağıt yoksa elini boşlukta sallayacaktır. Geribeslemenin genel prensipleri Yunanlılar ve Romalılar tarafından biliniyordu fakat geçen yüzyıla kadar kullanılmamıştır.

Yunanlıların, Romalıların ve eski Avrupalıların robot benzeri makineleri varsa da "robot" adını ilk ortaya koyan Çekoslovak piyes yazarı Karel Capek'tir. Çek dilinde "robota" kelimesi mecburi çalışan işçiye karşılık düşer. Capek 1921 yılında, R.U.R. (Rossum's Universal Robots) adında bir oyun yazdı. Bu oyunda robotlar, Rossum ve oğlunun, topluma hizmet için yarattığı insan görünüşlü yaratıklardı.

Dünyanın en iyi bilimkurgu yazarlarından biri olan Isaac Asimov ise robotlarla ilgilenen bilim dalına "robotik" diyen ilk kişidir. Robotlar hakkındaki hikayeleri, henüz olmayan fakat ilerde olması muhtemel sorunlarla ilgilidir.

Sayısal kontrol ve uzaktan kumandanın geliştirilmesiyle robotik büyük yol katetmiştir. Sayısal kontrol 1940 'ların sonlarına doğru, John Parson tarafından makine ekipmanları için geliştirildi Amerika Birleşik Devletleri Hava Kuvvetleri tarafından kullanıldı. Uzaktan kumanda ise 1940'larda Atom Enerjisi Komisyonu tarafından, radyoaktif maddeler üzerinde çalışırken kullanılmıştır. Bu teknolojiler iki kişinin çalışmalarıyla endüstri alanında kullanılmaya başladı.

Bunlardan ilki Cyril Walter Kenward 1954 Mart 'ında patentini aldığı, ilk endüstriyel robot denilebilecek bir sistem tasarladı.

Yukarıda bahsedilenlerden ikincisi Amerikalı George C.Devol'dur. Devol'un, günümüzün modern robotlarının ortaya çıkmasında başrolü oynayan iki icadı vardır. Bunlardan birincisi elektrik işaretlerini manyetik olarak saklayıp, bir makineyi kontrol etmek için bu kayıtları kullanan cihazdı. Doğuş tarihi 1946 olan cihazın ABD patenti 1952 'de alındı. İcatlardan ikincisine "Program Metni Transferi" adı verildi ve patenti 1961'de alındı. Devol'un patenti Kenward'dan birkaç yıl sonra alınmasına rağmen, modern endüstri robotunun babası Devol sayılır. Devol'un icadının İngiltere yerine Amerika'da bir sanayi oluşturmasının sebebi de Joseph Engelberger'in sentezci zekasıdır.

Joseph F. Engelberger 1949 yılında Columbia Üniversitesinin fizik bölümünden mezun oldu. Öğrenciyken Asimov'un etkileyici romanlarını okumuştu. 1950'lerin ortasında Stamford, Connecticut'taki bir şirketin uçak bölümünde baş mühendis oldu. Bu bölüm jet motoru kontrolu alanında çalışıyordu. Engelberger'in bu zamanlardan robotike ilgisi vardı. Bir tesadüf sonucu, Joseph Engelberger ve George Devol bir kokteylde tanıştılar. Devol, Engerberger'e program metni transferi buluşundan bahsetti ve ikili hemen bu icadın ticari sürümü üzerinde çalışmaya başladı. "**Unimate**" bu çalışmanın ürünü oldu. 1962 yılında Engelberger Unimation şirketinin başkanı oldu.

Bu tarihten sonra Unimation da dahil olmak üzere Amerika, Avrupa ve Japonya'dan pekçok şirket robotik üzerine uygulamalar geliştirdi. Robotik üzerine daha pekçok değerli katkılar oldu. Bunlardan biri Stanford Üniversitesi ve Stanford Araştırma Enstitüsü'nün geliştirdiği bilgisayar orijinli bir robot dili olan WAVE 'dir. 1974 yılında araştırma dili olan AL bunu takip etti. İlk ticari robot dili olan VAL, Unimation'dan Victor Scheinman ve Bruce Simano tarafından oluşturuldu. Bu dil ilk olarak Unimation' un **PUMA** (Programmable Universal Machine for Assembly) robotu üzerinde denendi. PUMA, nispeten kısa eklemli bir robottu ve temeli General Motors firmasının montaj otomasyonu çalışmalarına dayanıyordu.1979' da Japon Yamanashi Üniversitesi tarafından montaj amaçlı olan **SCARA** (Selective Compliance Arm for Robotic Assembly) geliştirildi. SCARA' nın pekçok ticari uygulaması 1981'den sonra piyasaya sürüldü.

Robotik teknolojisindeki çalışmalar artık büyük oranda bilgisayar teknolojisindeki gelişmelere dayanıyor. Robotik endüstrisi doğduğunda bilgisayarlar mevcut olmasına rağmen 70'lerin sonuna kadar boyutları nedeniyle robot kontrolunda kullanılmaya elverişli değildi. Bugün pazardaki tüm robotlar bilgisayar kontrolu kullanıyor. Fakat hala robotik alanının makine ekipman teknolojisi ve bilgisayar biliminin bir bileşimi olduğu düşüncesi yaygındır.

1.1 Niçin robot ?

Günümüzde, kullanılan robot sayısı yılda %35 oranında artıyor. Bu artışın sebeplerinden en önemlileri şunlardır:

1) **Üretim maliyetinin düşmesi:**
   a) Robotun bir saatlik işçilik maliyeti $5 iken ortalama bir işçininki yan ödemelerle beraber $20 'ı bulur.(Yan ödemelere sosyal sigorta, tatil ve hastalık izni, hastalık sigortası ve kıdem tazminatı dahildir. Oysa robot için bu ödemelere gerek yoktur.)

b) Kullanıldıkları zamanın %98'i oranında randımanlıdırlar. Oysa insanlar kahve molası, yemek molası ve kişisel ihtiyaçlar için işe ara verirler.

c) Robot kullanılarak toplam kalitede artış sağlanır, böylece maliyet düşer. Çünkü insanlardaki dikkatsizlik veya yorgunluk sonucu yapılan hatalara robotlarda rastlanmaz.

Genelde, bir robot alım bedeli, kurulma ve eğitim masrafları toplamını 12 ay içerisinde amorti eder.

2) **Üretim artışı:**

a) Bazı işlerde robotlar daha hızlıdırlar. Örneğin bir uygulamada kaynakçı bir robot, insanın harcadığı zamanın üçte birinde aynı işi tamamlamıştır.
b) Diğer bir uygulamada, araba boyayan robot 90 saniyede iç ve dış çift kat boyayı tamamlayarak günde 20 saat çalışırken aynı kaliteyi tutturmuş, en usta boyacı ise bu işi ancak 15 dakikada tamamlamıştır. (GM fabrikası Michigan)

3) **Kalitede artış:**

a) Robotlarda pozisyon doğruluğu insana göre çok yüksektir. Günümüzün robotları 0.003 cm doğruluğa ve 0.001 cm tekrarlanabilirliğe ulaşabilirler.
b) Çalışma hızı, yüksek kaliteli üretim için diğer bir avantajdır.

4) **Tehlikeli ortamlarda çalışma:**

a) Dökümhanede sıvı metal eriyiklerini boşaltma, zehirli boya ile boyama, radyoaktif ortamlarda çalışma gibi işler insanlar için çok zararlıdır ve robot kullanılması gereklidir.
b) Sualtı, uzay ve düşman bölgesinde araştırma çalışmaları gibi insana zarar verme ihtimali olan işlerde de robot kullanılması faydalıdır.

5) **Yönetim kolaylığı:**

Robotlar ön planlama sonuçlarına büyük doğrulukla ulaşabilirler. Ayrıca yaptıkları işi kaydedebilirler ve böylece işle ilgili istatistikler sağlanarak ileriye dönük planlar yapılabilir.

6) **Tümleşik sistem oluşturma:**

Robotlar, her yöneticinin hayali olan tam itaatkar ve denileni harfiyen yapan işçilerdir. Robotların yaptıkları işler, üretim düzeyi ve giren/çıkan mallar bir ana bilgisayardan izlenerek tam kontrolün sağlanabildiği tümleşik bir sistem oluşturulabilir.

7) **İş esnekliği:**

Sistemdeki bir hata nedeniyle çalışma değiştirilirse, karmaşık otomasyon sistemini değiştirmek yerine robot kolayca yeniden programlanabilir.

8) **Uzun ömür:**

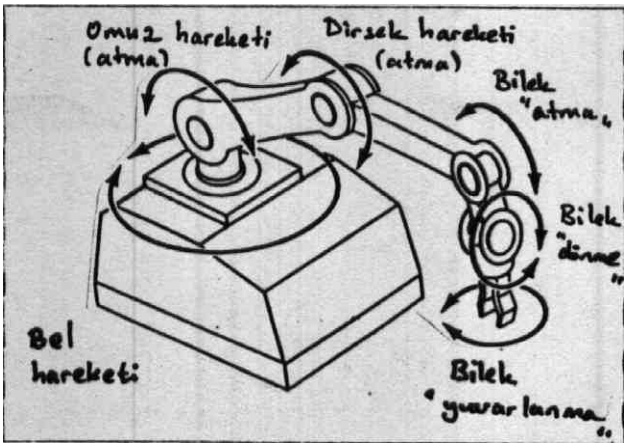
Uç elemanı değiştirilerek veya yeniden programlanarak modasının geçmesi engellenir, sistem ömrü uzatılır.

2. BİR ROBOTUN ELEMANLARI

Şekil 3.1'de bir robotun basit bir çizimi görülmektedir. Genelde, dışarıdan bakıldığında bir robotun bir taban ünitesi, bir kolu ve bir uç elemanı ya da kavrama elemanı bulunur. Uç elemanı genelde *çalışma aleti* adı verilen bir araç ya da parça taşır. Kol ya da taban ünitesinin içerisinde kolun hareketini sağlayan **sürücü** veya **motorlar** bulunur. Sürücülere gerekli kontrol işaretini vermek için bir kontrol cihazı ya da bilgisayar kullanılır. Robotun üzerinde ayrıca robotun kendi davranışlarını ve durumunu algılamak ve robotun dış ortamını algılamak için kullanılan duyargalar bulunur.

2.1 Linkler ve Mafsallar

Mafsalların yani bağlantı yerlerinin arasında kalan sabit kısımlar link adını alır. Mafsallar ise döner veya kayar şekilde olabilirler. Şekil 3.1'de tümü döner şekilde 6 mafsal görülmektedir. Mafsallara endüstride bir çok ad verilmektedir. Aşağıdaki şekilde en çok kullanılan isimleri görülmektedir.



Şekil 3.1. Bir revolüt robottaki eksenler ya da serbestlik dereceleri. Bileğin üç adet ekseni vardır : Atma (pitch), dönme (yaw) ve yuvarlanma (roll)

Linkler robotun cisimleri taşımasında destek olarak kullanılan katı ve haraketsiz elemanlardır. Genelde ağırlığı azaltmak ve vites aksamı, elektriksel kablolama, kontrol işareti hatları ve hava boşluklarına yer sağlamak için içi boş olarak yapılırlar. Linklerin yapımında *grafit fiber* malzemelerin kullanılmasıyla linklerin ve dolayısıyla robot kollarının ağırlıklarının oldukça azalması sağlanabilmektedir. Bu malzemeyle oluşturulan kollar çelik ya da alüminyum kollardan daha hafif ve de aynı derecede sağlamdır.

Kolun doğru şekilde konumlandırılabilmesi için mafsallar hassas olarak yapılmalıdır. Mafsalların doğru ve yumuşak şekilde hareket etmeleri için *bilyalı yataklar* kullanılır.

2.2 Bilek

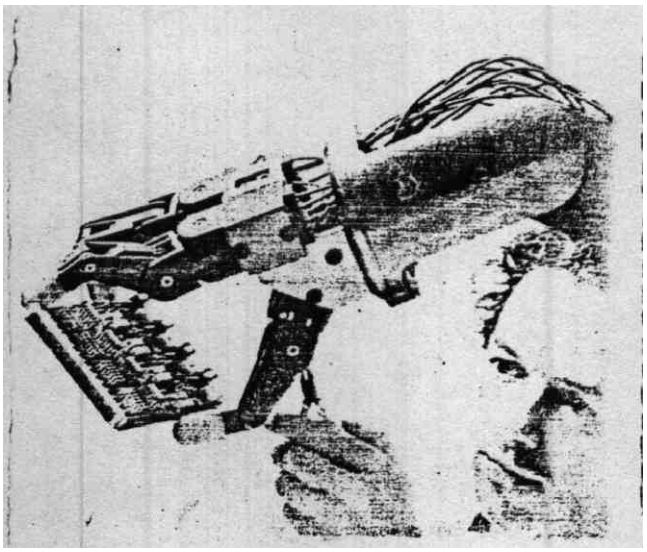
Bilekler kolun sonuna monte edilirler ve iki ya da üç bağlantı yeri (mafsal) içerirler. Genelde tek bir birim olarak yapılırlar. Bileğin üç hareketinin (dönme, yuvarlanma, atma) Şekil 3.1'de görüldüğü gibi ayrı merkezlerin çevresinden ziyade aynı merkez çevresinde olması istenir. Çeviklik ve kontrol edilebilme bu tek merkezli yapıda daha iyi sağlanabilir.

Tek merkezli yapıları mümkün kılan bazı dahice tasarımlar mevcuttur. Bileğin esnekliği küçük yerlerde robotun verimli ve doğru olarak iş yapabilmesinde büyük rol oynadığı için bilek tasarımı robot yapımında önemli bir unsurdur.

2.3 El ya da Uç Elemanı

Robot terminolojisinde kolun sonu bileğin de sonudur. Bileğin üzerinde bulunan bir yerleştirme elemanıyla bilek ile bir çok uç elemanının kullanılması mümkündür. Genelde kullanılan uç elemanları, kavrayıcılar, pençeler, kaynak tabancaları, havalı tutuculardır.

Yukarıda sayılan robot eli tipleri, belli uygulamalar için özel olarak oluşturulmuş ve genel amaçlı olmayan aletlerdir. Ancak son yıllarda yapay elle donatılmış ve her işi yapabilen robotların yapılmasına çalışılmaktadır. Yapay elin tasarımında, insan eli bir model olarak alınmaktadır. Zira insan eli, tabiatta bulunan en eşsiz mekanik alettir. Çok sayıda kastan oluşan değişik bir sistem tarafından yönlendirilen insan eli, birçok karmaşık işlemi kolayca gerçekleştirebilmektedir. Elde bulunan baş parmak, diğer parmakları idare eden bir görev yüklenmekte ve tutuşlarda kuvvet kavrayıcısı olarak görev almaktadır. Baş parmaksız bir eleman, normal bir elin ancak 2/3' ü kadar iş yapabilmektedir.



Şekil 3.2. Yanda Stanford Üniversitesi'nde gerçekleştirilmiş 3 parmaklı bir yapay el görülmektedir. 3 parmaklı yapı, karmaşıklığı azaltırken cisimlerin rahatça tutulmasını sağlıyor. Bu eli idare edebilmek için birçok motor, parmak uçlarında hassas algılayıcılar ve karmaşık bilgisayar programları gereklidir.

Son zamanlarda gerçekleştirilen robot ellerine bir örnek, 3 parmağı bulunan ve bir bilgisayarla yönetilen Darmstad Eli'dir. Bu elin tuttuğu bir cisim elinden alınmak istediğinde, sanki canlıymış gibi tepki göstermesi ve daha sıkı şekilde tutmaya çalışması üzerinde halen deneyler yapılmaktadır. Bir başka yapay el, yine biyolojik örneğe göre gerçekleştirilen, 5 parmağı ve parmaklarının uçlarında bir çok duyarga olan Utah/MIT eli'dir.

İnsan elini taklit eden bu yapay ellerin hepsinin parmakları vardır. Ancak bu parmakları insan elinde olduğu gibi kaslar değil, *elektrik motorları* harekete geçirmektedir. Bu nedenle yapısal olarak biyolojik ellerden büyük farklar gösterirler ve daha çok yer kaplayarak daha az karmaşık işleri gerçekleştirebilirler. Örneğin Darmstad eli'nin parmaklarına motor gücü üzeri sarılı bir çelik tel kablosu (Bowden kablosu) yoluyla gelir ve elin arka kısmında elektromotorlardan oluşan bir blok bulunur. Çelik tel ve üzerinde sarılı kılıf arasında meydana gelen sürtünme kuvveti, üç parmak arasında oransızlığa ve çekiş gücünün bozulmasına neden

olur. Buna karşılık çok sayıda sinir ve kasa sahip insan kolu, elin yumuşak şekilde hareketini sağlamaktadır.

Geliştirilen bir başka el olan Karlsruhe eli'nde ise çelik tellerle güç aktarımı yerine elektromotorlar doğrudan bağlantı yerlerine konulmuşlardır. Bu yapı robot elinin biyolojik yapıdan tamamen uzaklaşmasına ve çok ağır ve hantal hale gelmesine neden olmaktadır.

Son zamanlarda Amerikan, İngiliz ve Japon bilim adamları tarafından termik, kimyasal ve elektriksel olarak uyarılabilen *plastikten yapılmış kaslar* geliştirilmiştir, ancak halen deneme safhasındadır. Ayrıca *şekil bellekli alaşımlar* da kullanılmaya çalışılmaktadır. Bunlar şekilleri ısıya bağımlı metal alaşımlardır ve belli bir sıcaklığa getirildiklerinde önceden belirlenen bir konuma kadar parmağın eğilmesi sağlanabilmektedir. Bochum-Ruhr Üniversitesi'nde ince tel parmakları bellekli alaşımlardan yapılmış küçük bir robot eli modeli halen deneme aşamasındadır.

Robot elinin istenen fonksiyonları karşılayabilmesi için, mekanik yapı kadar, robotu kontrol etmekte kullanılan bilgisayarlar ve programlar da büyük bir öneme sahiptir. Zira beş parmaktan aynı anda gelen duyarga bilgilerinin alınması, işlenmesi ve her parmağın uyum içerisinde hareket ettirilmesi gereklidir. Bunun için birbiriyle paralel olarak çalışan programlar ve çok işlemcili mimariler kullanılmaktadır.

2.4 Sürücüler ve Sürücü Mekanizması

Linkleri bağlantı yerlerinin etrafında hareket ettirmek için elektrik motorları, hidrolik motorlar, hidrolik silindirler ve pnömatik motorlar kullanılır. Bilyalı vida, vites ve harmonik dişli gibi çeşitli mekanizmalar yardımıyla sürücü hareketi istenen hız ve yöndeki bir harekete dönüştürülür.

Robotların ömrü ve güvenilirliği, vites, yatak ve bilyalı vida elemanlarındaki gelişmelerle artmıştır. Bu konudaki en büyük gelişme belki de *harmonik dişlilerdir*. Harmonik dişliler 1/100 ve daha fazla oranlarda hız düşürme işlemini büyük bir doğrulukla ve küçük bir hacimde gerçekleştirebilirler. Harmonik dişlilerle klasik vites sistemlerindeki doğruluğu azaltan geri tepme sorunu da ortadan kalkmaktadır.

Unimate gibi ilkel robotlarda sürücü olarak hidrolik silindirler ve hidrolik dönen motorlar kullanılmıştı. Daha sonraları güç tranzistörlerindeki gelişmeler ve az bulunan güçlü manyetik malzemelerin kolay olarak üretilmesiyle, motorlar ve bunların kontrolünde kullanılan devreler hem daha az yer kaplar hem de daha ucuz imal edilir hale geldi. Bu nedenle artık elektrikli motor ve kontrol yöntemleri kullanılmaktadır.

Katı hal fiziğindeki devrimin bir başka sonucu fırçalı DC motorlar yerine fırçasız olanlarının kullanılmasıdır. Kommütatör ve onlarla elektriksel teması sağlayan fırçalar kısa sürede yıpranmakta ve temas sonucu oluşan arklar nedeniyle gürültü kaynağı oluşturmaktaydılar. Bu da motor ömrünün kısalmasına sebep oluyordu. Daha pahalı olmalarına rağmen *fırçasız motorlar* bu yıpranma kaynağını ortadan kaldırmış ve güvenilirliği arttırmışlardır.

Bir başka yenilik, DC motorlardan daha güvenilir olan yönü değiştirilebilen *AC servo motorların* kullanımıdır. Robotlarda ayrıca *adım motorları* da kullanılmaktadır.

Sürücüler ile birlikte kullanılan sürücü mekanizmaları şu şekilde sayılabilir :

Mekanizmalar :

- Doğrusal Hareket Mekanizmaları
- Kayan dişli sistemi
- Dişli kayış ve zincir düzeni

- Bilyalı vidalar

Dairesel Hareketler

- Dişli kutusu
- Harmonik dişli
- Sonsuz Vida

2.5 Kontrol Mekanizmaları

Kontrol mekanizmaları deyimi, kollar, bilekler ve uç elemanlarını uyumlu bir şekilde hareketini sağlamak ve yönlendirmekte kullanılan tüm cihazları içine alır. Basit bir kontrol düzeneği hava vanalarını açıp kapamakta kullanılan bir zamanlayıcı devreden ibaret olabilir. Kaldırma ve yerleştirme işlerinde kullanılan bir robotta, hareketi sınırlamak için mekanik durdurucular kullanılabilir. Daha karmaşık kontrol sistemleri adım anahtarlarını, bilgisayar ve lojik devreleri ya da robot linklerini süren servo motorların kontrolünde kullanılan mikroişlemci sistemlerini içerebilir.

Mikroişlemcilerle birlikte roborlarda kullanılan kontrol yöntemleri tümüyle değişmiştir. Bugün birçok robotta her bağlantı yeri için bir mikroişlemci ve de tüm kolu ve dış dünyaya olan arabirimi kontrol etmek için bir ana bilgisayar kullanılmaktadır. Robotlarla kullanılan görme ve duyarga sistemleri kontrol sistemlerinde, bir ya da daha fazla mikrobilgisayar ve hatta minibilgisayarlar içermektedir. Robot kolunun düzgün bir şekilde hareketi için artık paralel mimariler kullanılmaya başlanmıştır.

2.6 Duyargalar

En basit ardışıl makinalar dışında tüm robot sistemlerinde bağlantı yerlerinin konumunu ölçmek için konum duyargaları kullanılır. Kaydı tekrarlayan robotlarda bağlantı yerlerindeki duyargalardan gelen konum bilgileri kaydedilir ve gelecekte kullanılmak üzere depolanır. Her bağlantı yerindeki depolanmış konum bilgisi, robot önceden kayıt edilen işi gerçekleştirirken bağlantı yerindeki sürücüleri kontrol etmekte kullanılır.

Duyargalar ayrıca kuvvet, tork, yakınlık, sıcaklık vb. ölçümlerinde de kullanılırlar. Son 5 yıl içerisinde robot pazarında birçok yeni duyarga çeşidi ortaya çıkmıştır. Parçaları birleştirme işlemlerinde kullanılmak üzere üç eksende kuvvet ölçümü ve üç eksende de tork ölçümü yapabilen bir bilek duyargası kullanılabilmektedir. Küçük parçaları yüzeysel özelliklerine göre dokunma yoluyla tanımada kullanılan dokunma duyargaları da artık kullanılmaktadır. Duyargalarda meydana gelen bu ve daha bir çok gelişme sonucu günümüzde robotlar, radyoaktif maddelerin tanınarak temizlenmesinden bombalı paket imhasına, cerrahi operasyonlardan uzay araştırmalarına bir çok alanda pratik olarak kullanılabilir duruma gelmiştir.

2.7 Arabirimler

Arabirimler, robotun tüm amaçları için dış dünyayla olan bağlantılarıdır. Yardımcı cihazlar, bilgisayarlar ve dış duyargalardan elektriksel işaretlerin alınması gereklidir. Genelde robotun yeni bir göreve başlayabilmesi için bir işin ya da hareketin tamamlandığını bildirmesi istenir. Nümerik kontrollü bir tezgahın gerektiğinde açılarak robotun tezgah içerisine bir cisim yerleştirmesine olanak vermesi istenir. Bu durumda tezgahın açılıp robotun cismi koyması için durduğunu belirten bir işaret robota gönderilmelidir. Robot parçayı yerleştirdikten sonra, parçayı koymak üzere tutacağını bırakmalı ve tezgaha işlemin tamamlandığını elektriksel bir

işaretle bildirmelidir. Robotlarla diğer elektronik cihazların haberleşmesinde RS-232 gibi standartlar kullanılmaktadır. Görme uygulamalarında ve televizyon taraması için RS-170 standardında arabirimler kullanılır.

3. ROBOT PROGRAMLAMA DİLLERİ

Manipulatörlerin genel amaçlı makina montajı olarak kullanılmasındaki baş engel, kullanıcının verilen görevi tamamamlamak için manipulatörü yönetmesi için gereken uygun ve etkili iletişimi sağlanamamasıdır. Robotla iletişimi sağlamak için bazı yollar vardır, bunu elde etmek için ise 3 ana yaklaşım vardır; ayrı kısımlardan ibaret kelime tasdiği, öğretim ve playback ve yüksek sınıf programlama dili.

Kelime tasdiği sistemleri, limitli bir sözlükten bir grup ayrı kısımlardan ibaret kelimeleri tanıyabilir ve genellikle kullanıcının kelimeler arasında kendisini durdurmasını talep eder. Şimdi ayrı kelimeleri, hızlı bilgisayar unsurlarından ve etkili yöntem algoritmalarından dolayı gerçek zamanında tanımak mümkün olsa bile, bir robot görevini tanımlamak için ayrı kelime tasdiğinin kulllanılabilirliği faaliyet alanında biraz sınırlandırılmıştır. Dahası, lisan tanınması genellikle geniş bir bellek veya bilgi lisanın depolanması için ikinci bir depo talep eder. Ayrıca konuşma şeklinin anlaşılabilmesi için önce bir deneme peryodunu da talep eder.

Öğretim ve playback-"guiding" olarak da bilinir ve günümüzdeki endüstriyel robotlarda en fazla kullanılan metottur. Bu metod, kullanıcının robotun gerçekleştirmesini istediği hareketleri kumanda ederek robota öğretimi de içerir. Öğretim ve playback, tipik olarak şu adımlarla tanımlanır:

1. Bütün gürevin montajında el kontrolü kullanarak robota yavaş harekette rehberlik etmek ve hareketin tekrarı için belirli konumlarda robotun birleşik açılarını kaydetmek,
2. Düşünülen hareketi yayımlamak ve playback yapmak,
3. Eğer düşünülen hareket doğruysa, tekrarlamalı modda ve belirli bir hızda robot çalıştırılır.

Robota yavaş harekette rehberlik etmek çeşitli yollarla elde edilebilir: joystick kullanımı, bir gurup basım düğmesi (her birleşim için bir tane) ya da master-slave(efendi-köle) manipulatör sistemi. Şimdilerde, en sıkça kullanılan sistem, basım tuşlu el kutusudur. Bu metodda kullanıcı robotu eliyle mekanda hareket ettir ve manipulatörün istenilen açısal pozisyonunu kaydetmek için tuşa basar. Kaydedilen açısal pozisyon gurubu manipulatörün çaprazladığı yörüngenin grup noktalarını meydana getirir. Bu pozisyon grup noktalarında, numaralı metodlarla ara değer bulunur ve robot düzeltilmiş yörüngenin içinden playback yapılır. Yayımlanan playback modunda, kullanıcı kaydedilmiş açısal pozisyonları yayınlayabilir ve robotun görevini tamamlarken çarpmayacağından emin olur. Runé modunda robot yayımlanmış ve düzeltilmiş yörüngesine göre tekrar tekrar çalışacaktır. Eğer görev değiştirilirse yukarıdaki 3 aşama tekrar edilir. Bu metodun avantajları öğreniminin kolay olması ve açısal pozisyonların kaydı için nispeten daha ufak bir alan talep etmesidir. Bu metodun ana dezavantajı da kontrol sistemi içinde bu metoddan duyumsal geri besleme bilgilerini tamamlamak için faydalanılamamasıdır.

Yüksek sınıf programlama dilleri, insan-robot iletişim problemlerinin çözümünde daha genel bir yaklaşım sağlarlar. Robotlar yay kaynak yapımı ve "guiding" (Engelberger 1980) kullanarak spreyle boyama alanlarında hiçbir şekilde birbirine tesiri talep etmezler ve "guiding" sayesinde kolaylıkla programlanabilirler. Bununla beraber, robotların montaj görevi

için kullanımı yüksek sınıf program teknikleri talep eder çünkü robot montajı genellikle duyumsal itilim bilgilerine güvenir ve bu tip planlanmamış birbirine tesir sadece şartlı programlanmış metodlarla ele alınabilir.

Robot programlaması esasen geleneksel programlamadan farklıdır. Biz herhangi bir robot programlamasıyla ele alınması gereken birkaç düşünce tanımlayabiliriz:

Robot tarafından el ile işletilecek nesneler 3 boyutlu ve değişik fiziksel özellikleri olan nesnelerdir. Robotlar uzaysal kompleks çevrede yönetirler; 3 boyutlu nesnelerin bilgisayar içinde açıklamaları ve ifade edilişleri kesin değildir ve duyumsal bilgiler izlenmeli, el ile işletilmeli ve doğru bir şekilde kullanılmalıdır. Programlamaya yönelik şimdiki yaklaşımlar iki ana kategoride sınıflandırılabilir. Robota yönelik programlama, nesneye yönelik veya görev sınıf programlama.

Robota yönelik programlamada, bir montaj işi robot hareketlerinin ardışıklığı olarak açıkça tanımlanır. Robota bütün görev içinde programın her durumunu kabaca robotun bir hareketini karşılayacak biçimde program tarafından kontrol ve rehberlik edilir. Diğer yandan, görev-sınıf (task-level) programlama montaj görevini, robotun ardışıklığı olarak tanımlar ve bundan dolayı hiçbir açık robot hareketi özelleştirilemez.

3.1 ROBOT-SINIF DİLİNİN ÖZELLİKLERİ

Robot-sınıf dilinin dizaynında alınan en yaygın yaklaşım robot programlama isteklerini karşılamak için varolan yüksek sınıf lisanının genişletilmesidir. Nesnelerin çalışma alanı içersindeki konumları ve boyutları sadece belirli bir doğruluk derecesine kadar açıklanabilir. Bir robotun bu tip belirsizliklerin varlığında görevlerini yerine getirmesi için hassasiyet uygulanmalıdır. Toplanan duyu bilgileri aynı zamanda robotun çevresindeki kuruluşun durumunu denemesi ve doğrulamasını sağlayan bir çevreden geri besleme olarak da rol alır. Robot programlamasındaki hassasiyet 3 tipte sınıflandırılabilir:

1. Pozisyon hassasiyeti robotun geçerli pozisyonunu belirlemek için kullanılır. Bu birleşik açıları ölçen ve çalışma alanındaki el pozisyonlarını hesap eden kodlayıcı tarafından yapılır.
2. Kuvvet ve dokunma hassasiyeti çalışma alanındaki nesnelerin varlığını tespit etmekte kullanılır. Kuvvet hassasiyeti istenmeyen hareketlerde güç kontrollü hareketleri sağlamaya yönelik geri beslemeye dönmek için kullanılır. Dokunma hassasiyeti ise nesneyi kavrarken kayganlığı tespit etmekte kullanılabilir.
3. Görüntü nesneleri tanımlamak ve onların pozisyonlarına dair kaba bir tahmin sağlamakta kullanılır.

Hassasiyet komutlarının nasıl işletileceğine dair sanal bir konsensüs olmayıp her dilin kendine has yazılımı mevcuttur. AL güç hassasiyeti için FOCE(axis) ve TURQUE (axis) gibi primitivleri sağlar. AML farklı zamanlı olayları tespit etmek için hareket komutlarında özelleştirilebilecek MONITOL olarak isimlendirilen bir primitiv temin eder. Çoğu diller vizyonu kesin olarak desteklemez ve kullanıcı vizyon bilgilerini elde etmek için modüller temin etmek zorunda kalır.

Biz bir robotun programını meydana getirmede bulunan adımları deneyerek bütün robot dillerinde yaygın olan bazı anahtar karakterini hatırlayabiliriz. Sürgüyü deliğin içinden geçirme işini düşününüz. Bu robotu besleyiciye hareket ettirmeyi ve deliklerin birisinin içinden geçirmeyi gerektirir. Tipik olarak programı geliştirmek için atılan adımlar şunlardır:

1. Çalışma alanı kurulur ve bölümler, fikstürler ve besleyiciler kullanılarak sabitleştirilirler.
2. Parçaların (beam , feeder vb...) konumu (yönlendirme ve pozisyon) ve onların özellikleri (bam -bore , bolt- grasp vb...) lisan tarafından sağlanan bilgi yapılarını kullanarak tanımlanır.
3. Bu birleşme çabası robotu hareket ettirmek, nesneleri kavramak ve itimi gerçekleştirme gibi faaliyetler içinde bölüştürülür.
4. Duyum komutları anormal durumları (boltu kavrarken yerleştirme zaafiyeti gibi....) tespit etmek ve birleştirme görevi sürecini bir yerde toplamak için eklenir.
5. İkiden dörde kadar ki adımların tekrarlanması suretiyle programın akışı sağlanır.

AL, Stanford Ün. tarafından geliştirilmiştir. Kolların gerçek zaman kontrolü sadece POP 11 standında uygulanır. Onun karakteristik özellikleri şunlardır.

- ALGOL ve PASCAL özellikli yüksek dil seviyesi
- Robot-level ve task-level özelleştirilmesini destekler.
- Senkronizasyon concurent execution ve on condition gibi ... programlama dil yapısına sahiptir.
- Geniş bir kullanımı olmasını sağlar. AML IBM tarafından geliştirilmiştir. IBM RS 1 robotunun kontrolü içindir. RS 1, 6 serbest dereceli bir kartezyen yöneticisidir.

3.1.1 Pozisyon Özelleştirilmesi

Robot montajında, robot ve parçaları genellikle iyi tanımlanmış bir alanla sınırlanırlar. Bu bölümler pozisyonel kararsızlıkları en aza indirgemek için beslenici ve fikstürler tarafından genellikle sınırlandırılır.

Rasgele yerleştirilmiş bir grup parçanın montajı görüntü talep eder ve endüstride henüz yaygın olarak kullanılmaz.

Nesnelerin çalışma alanındaki yöneltme ve pozisyonlarını tanımlamada kullanılan en yaygın yaklaşım, koordinat çerçevesiyle olanıdır. Onlar genellikle 4x4 homojen dönüşüm matrisleri olarak temsil edilirler. Bir çerçeve 3x3 alt matristen meydana gelir. AL tarafından alınan yaklaşımlar, FRAME, dönüşsel matrisler (ROT) ve vektörler (VECTOR) adlı çerçeveler için önceden tespit edilmiş bilgi yapılarını sağlamak içindir (hepsi kartezyen koordinatları şeklinde). Yukarıda sözü edilen çerçeve yapıları hakkında bilgi edinmek için PL'deki ilk durum taban koordinat çerçevesinin saptanması gerekir. Bu koordinat çerçevesinin eksenleri referans çerçevesinin baş eksenlerine paraleldir. Bu çerçevenin başlangıcı referans çerçevesinin başlangıcından (20,0,15) inç uzaklıktadır. AL'daki ikinci durum beam koordinat çerçevesini belirler. Bu beam çerçevesinin ara yörüngesi referans çerçevesinin z yörüngesini 90 derece döner ve bu beam çerçevenin başlangıç noktası referans çerçevenin başlangıç noktasından (20,15,0) inç uzaklıktaki bir konumdadır. 3. durum birincisiyle aynıdır ancak konum birinciden farklıdır. AML'deki üç durumun anlamları AL 'deki durumların anlamlarıyla aynıdır.

Homojen transformasyon matrisi kullanmanın bir avantajı da, baz çerçeveyle bağıntılı bir çerçevenin tanımlanmasının transformasyon matrisinin, baz çerçeveyle sonradan-çarpma yaparak çok kolay bir şekilde elde edilebilmesidir. AL bir matris çarpım yöneticisi sağlar.

İlk AL durumu koordinat T6 çerçevesinin temsili demektir. AL durumundaki bu T6 koordinat

çerçevesinin ana yörüngeleri baz koordinat çerçevesinin X yörüngesinin etrafında 180 derece döndürülmüştür ve bu koordinat çerçevesinin başlangıç noktası baz çerçevesinin başlangıç noktasından (15,0,0) inç uzaklıktadır. İkinci durum E koordinat çerçevesini temsil eder. Bu koordinat çerçevesinin ana yörüngeleri T6 koordinat çerçevesine paraleldir ve E koordinat çerçevesinin başlangıç noktası T6 koordinat çerçevesinin başlangıç noktasından (0,0,5) inç uzaklığındaki bir konumdadır. Diğer üç AL durumuna da benzer komutlara başvurur. AML durumlarının anlamları da AL durumlarının anlamlarıyla aynıdır.

Nesnenin yönlendirilmesini ve pozisyonunu elde etmenin bir diğer yolu da robotu bilgiyi etkileşimli olarak almayı sağlayan bir işaret cihazı olarak kullanmaktır. POINTY, AL için dizayn edilmiş bir sistemdir. Kullanıcının çalışma sahasında robotu yönetmesine izin verir (elle veya pedallarla) ve POINTY, AL tanımlamalarını işaret edebilir ve özel bir aletle çizer. Bu çerçeveler arasındaki açıları ve uzaklıkları ölçmedeki ihtiyaçları sağlar.

Koordinat çerçeveleri her ne kadar robot konfigürasyonlarını temsil etmede çok kullanılsa da bazı sınırlandırmalara sahiptir. Nesneler çoğaldıkça koordinat çerçeveleri arasındaki bağıntılarda giderek karışık bir hale dönüşür ve yönetilmeside o denli güçleşir.

4. YAPAY ZEKA

Görüş ve hassas algılama gibi robot kontrolünün modern konuları, yapay zekadaki (Artificial Intelligence-AI) gelişmelerle ortaya çıkmıştır. Yapay zeka, zekice davranan sistemler geliştirmeye çalışır. Bu, "düşünen makineler" i hedefler. Bu dal her ne kadar bilgisayar biliminin bir alt kolu olsa da, ayrıca psikoloji, dilbilim ve de matematikle de ilgilenir.

Yapay zeka, bilgisayar biliminin bir dalı olup, insan davranışlarındaki zeka (dili anlama, öğrenme, mantıklı davranma ve problem çözme) ile ilgilenir. Zekanın tanımının güç olmasından dolayı, yapay zeka en iyi hedefleri ile tanımlanabilir.

4.1 YAPAY ZEKA ARAŞTIRMALARININ HEDEFLERİ

1. **Problem Çözme:** Problem çözme sistemlerine örnek, satranç programlarıdır. Bu sistemler, verilen kurallar ve stratejilerle uzman denilebilecek seviye de satranç oynama yeteneğine sahiptir. Problem çözme oyunlarla sınırlı değildir. İlerde evinizdeki robota bir kağıdı alıp getirmesini emrettiğinizi düşünün. Bu kağıdı alıp getirme işlemi sırasında robotun bir çok küçük problemi uygun sırada çözmesi gerekir. İlk olarak kağıda erişmek için uygun bir yol geliştirmelidir. Daha sonrada bu yol üzerindeki engellerle ilgilenmelidir (kapıyı açmak vb.). Son olarak da kağıdı kavrayıp getirmelidir. Günlük hayatımızda karşılaştığımız çoğu iş esasında bir çok problemi çözmeyi gerektirmektedir. Gerekli sistemlerin tasarımı, problemlerin, çözümler görünür olana kadar, daha küçük problemlere bölünmesini gerektirmektedir. Bu noktada bizim robota bir montaj işlemini gerçekleştirebilmesi için tüm hareketleri "göstermemiz" gerekir. Bir problemin çözümü için geliştirilmesi gereken teknikler, robotun montaj stratejisi ile aynıdır.
2. **Doğal Dil:** Her ne kadar bilgisayar hayatımızda artık çok önemli bir yer tutsa da, yine de büyük bir çoğunluk bilgisayarlara alışma konusunda büyük zorluk çekmektedir. Bunun nedeni ise bilgisayarlarla, kendi "doğal dilimizle" değil de bilgisayarın diliyle iletişim kurmak zorunda olmamızdır. Doğal dil araştırmacılarının karşılaştıkları sorunlar çok fazladır. Bilgisayarlar yalnızca kelimelerin anlamını bilmekten öte, kelimelerin konuya göre nasıl değiştiğini de fark edebilmelidir. Ayrıca bilgisayarlar, kelimelerin birbirleri ile ilişkisini anlayabilmek için dilin sözdizimini de anlayabilmelidir.
3. **Uzman Sistemler:** Araştırmaların bu kolu, özel alanlarda uzman kişiler gibi davranabilen sistemlerin geliştirilmesi ile ilgilenir. Uzman bir sistem, bir operatörle yapılan diyalog yoluyla, karara varılana kadar uygun sorular sorulmasını veya testler yapılmasını sağlayabilir. Şu anda uzman sistemler, bilgisayar sistemlerinin tasarımında, devrelerin dizaynında ve de tıbbi teşhislerde kullanılmaktadır. Uzman sistemlerin tasarımı, karara varılması, uzman kişilerden alınan verilerin sunulması, deneyimle "bilgi tabanının" geliştirilmesi yolunda, büyük miktardaki verileri işleme ve de mantıklı davranma gibi problemlerle karşılaşır.
4. **Öğrenme:** Zekanın bir özelliği de, deneyim yoluyla öğrenebilmedir. Eğer makineler öğrenme yeteneğine sahip olsalardı o zaman uzman sistemler konusunda karşılaşılan, bilgi sağlama işi büyük ölçüde kolaylaşacaktı. Deneyimle öğrenebilme yeteneğini gösteren bazı sistemler geliştirilse de, bu zamana kadar çok sınırlı bir ilerleme kaydedilmiştir.
5. **Görüş:** Yapay zekanın görme konusundaki en ilginç hedefi, sistemlerin görüntü analizi yapmalarını sağlayabilmektir. Bu ise, görüş sistemini bir görüntü ile sunması ve de sistemin bu görüntü içindeki nesneleri belirleyebilmesi demektir.

2.

4.2 YAPAY ZEKA TEKNİKLERİ

Yapay zeka, bilginin veya verilerin kullanılması ile ilgilenir. Bu yüzden teknikler iki temel konu üzerinde geliştirilmelidir: verilerin ifade edilmesi ve de verilerin yönetilmesi.

4.2.1 Bilginin İfade Edilmesi

Bilginin ifade edilmesi konusu, bilgisayarın fiziksel işlemleri ile ilgili değildir. Bu konu, gerçeklerin birbirleri ile ilişkisini ele almaktadır (örnek olarak, "Bazı kuşların kanatları vardır." ifadesi).

Bilginin, çok çeşitli olan ifade ediliş yöntemlerini tartışmadan önce, ifade edilmeye ihtiyaç duyabilecek bilgi türlerinin tanımlanması gerekir.

1. **Nesneler:** "Kuşların kanatları vardır" gibi, nesneler hakkındaki daha spesifik gerçekler.
2. **Olaylar:** "Robot öğrencisi robotun kolunu kırdı", gibi yalnızca olayın kendisinin ifade edilmesinden öte, "Robot öğrencisi dün robotun kolunu kırdı ve de acımasız asistan bunu ödemesini istedi.", gibi zaman veya neden- etki ilişkisini içeren ifadeler.
3. **Performans:** Eğer yapay zeka sistemi bir robotun kontrolü için tasarlanmışsa, kinematik, dinamik, yönetmek için donanımda hangi bitlerin kullanılacağı ve bunun gibi robot koluna ait performans verilerine sahip olması gerekir.
4. **Metabilgi:** Bu bizim bilgimiz hakkındaki bilgidir. Bizim bilgimizin kökenini, bağıl önemini, güvenilirliğini ve bu tür niteliklerini içermektedir. Mesela bazıları "Robot dersi zor değildir." bilgisine, eğer bu bilgi bir tarih öğrencisinden geliyorsa, pek önem vermeyebilir.

Bu aşamadan sonra bilginin ifade edilmesi tekniklerini inceleyebiliriz.

1. **Mantık:** Biçimsel mantık, gerçeklerden sonuç çıkarmayı sağlayan bir metot oluşturulması için, matematikçiler ve filozoflar tarafından geliştirilmiştir. Biçimsel mantık, gerçeklerin spesifik bir sözdizimi içinde ifade edilmesini ve de tanımlanan sonuç çıkarma kurallarını gerçeklere uygulayarak, karara varabilmeye izin verir.

3.

a) Bütün robot bilimciler oyun oynar.
b) Jack bir robot bilimcidir.

Yukarıdaki ifadelerden:

c) Jack oyun oynar.

sonucunu çıkarabiliriz.

Belki bir bilgisayarda, bir konu hakkındaki tüm bilgiler ve tabii ki sonuç çıkarma konusunda uygulanabilen tüm kurallar da verildiğinde, konunun orijinalinden yeni gerçekler geliştirebilir. Bu konu, mantığa dayalı tüm ifade biçimlerini sınırlayabilecek, "Kombinasyonal Patlama" görüşü altında incelenmektedir. Gerçeklerin sayısı arttıkça, gerçeklerin uygulanabilecek sonuç çıkarma kuralları ile oluşan kombinasyon sayısı da artmaktadır. Bu ise, bilgisayarın uygun bir süre içinde çözemeyeceği kadar büyük bir sorun oluşturmaktadır.

2. **Yönteme Dayalı İfade:** Gerçeklerin saklanması kadar, gerçeklerin nasıl kullanılacağı ile ilgili bilgilerin saklanması da önemlidir. Yukarıda anlatılan mantığa dayalı sistemlerde, sonuç çıkarma ile ilgili kuralların tümü yeni bir noktanın ispatı için uygulanmalıdır. Eğer bilgileri kullanımları ile ilgili olarak şifreleyebilseydik, daha verimli bir sistem geliştirilebilirdi. Örneğin, Jack'in oyun oynayıp oynamadığını bulmak isteseydik, bilgisayarın veritabanı şu gerçekleri içerebilecekti:

a. Jack bir robot bilimcidir.
b. Bütün robot bilimciler oyun oynar.

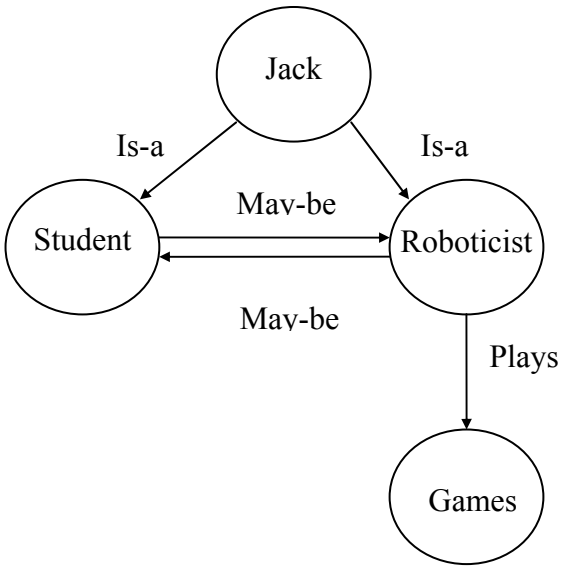
Ek olarak sistem, Jack hakkında aşağıdaki yöntemi de saklamış olabilir:

If need-prove plays (x)
show is-roboticist (x)

Bu yöntem, eğer x hakkında bir gerçeği ispatlamak istiyorsak, bunu x'in bir robot bilimci olduğunu göstererek yapabileceğimizi ifade eder. Bu bilgi ile sistem, Jack hakkındaki diğer tüm bilgileri göz ardı ederek, hemen doğru çözüme varacaktır. Bilginin yöntem olarak saklanması, bilgisayar programlarında if-then ifadesi ile karşımıza çıkmaktadır. Bu, sistemi doğrudan cevabı bulmaya yönlendirir. Ama bu yöntem, sistemi genellikten uzaklaştırır; değişiklik yapmayı zorlaştırır.

3. **Anlamsal Ağlar:** Anlamsal ağlar, düğümler ve yaylar içeren, bilgi ifade şeklidir. Tipik olarak düğümler, nesneleri, kavramları veya durumları; yaylar da düğümler arasındaki ilişkiyi gösterir. Düğümler ve bağlantılar basit dil tanımları ile etiketlenir. Şekil 1. Aşağıdaki bilgileri anlamsal ağlarla gösterir.

Jack bir öğrencidir.
Jack bir robot bilimcisidir.
Robot bilimciler oyun oynarlar.
Robot bilimciler öğrenci olabilirler.
Öğrenciler robot bilimci olabilirler.



Ağ incelenerek, Jack, öğrenciler ve de robot bilimciler hakkında bir çok sonuca varılabilir.

Anlamsal ağlar, yapay zeka araştırmalarında yoğun olarak kullanılır. Ama bunların da dezavantajları vardır. Çok basitleştirici olabilirler: Bir ağ fikirleri ve çok sayıdaki bilgiyi nasıl işleyebilir? Bütün ifade tekniklerinde olduğu gibi bu da bir yere kadar ilerleyebilir.

4. **Üretim Sistemleri:** Üretim sistemleri bilgiyi "üretimler" olarak adlandırılan parçalar halinde saklarlar. Üretimler, "IF + bazı ifadeler, doğruysa, THEN + bazı faaliyetler" şeklinde

bir yapıya sahiptir. Jack ve robot bilimciler hakkında sahip olduğumuz bilgiler şu şekilde sunulacaktır.

a. EĞER kişi bir robot bilimci ise, O ZAMAN o oyun oynar.
b. EĞER kişi bu kitabı okuyorsa, O ZAMAN o bir robot bilimcidir.

Eğer Jack'in bu kitabı okuduğunu görseydik, o zaman onun, gerçekten de bir oyuncu olduğunu bilecektik.

Üretim yöntemleri, problemi sıra ile parçalara ayırarak daha yönetilebilir işler haline getirir. Bunlar "uzman sistemlerin" klasik yapısıdır. Bu sistemler IF-THEN yapısını kullanarak düzgün tasarlanmış sistemler sağlarlar. Ayrıca her kuralın diğer kurallara doğrudan bir etkisinin olmadığı modüler bir yapı oluştururlar. Ama diğer problemlerin yanında, bu sistemler büyüdükçe verimsizleşmeye başlar.

5. **Çerçeveler:** Diğer bir ifade şeklide çerçeve kullanımı ile gerçekleşir. Çerçeveler, spesifik gerçeklerle doldurulacak boş yerlere sahip, önceden tanımlanmış yapılardır. Mesela bir öğrencinin genel çerçevesi aşağıdaki gibidir.

ÖĞRENCİ Çerçevesi
Boy: (cm)
Ağırlık: (kg)
Dersler: (Robot bilimi veya Tarih)

Jack için bu çerçeve şu şekilde olacaktır.

JACK Çerçevesi:
Boy: 1.70 cm
Ağırlık: 68 kg
Dersler: Robot Bilimi

Boş yerler diğer çerçeveler referans alınarak veya boşlukların nasıl doldurulacağına ait yönteme uyularak yapılır.

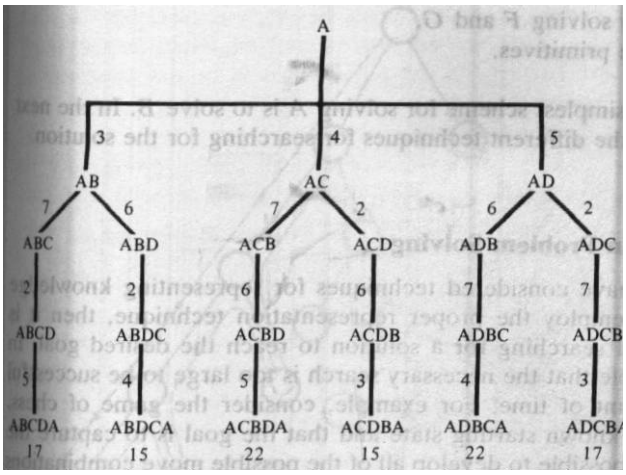
6. **Diğer İfade Teknikleri:** Çoğu durumda doğru ifade tekniğini seçmek, problemi oldukça basitleştirir. Örnek olarak bir nesnenin video görüntüsünü inceleyelim. Anlamsal ağlarla, her bir piksel bir nesneyi ve her bir yay, bitişik düğümlerin "-e bağlı" özelliğini gösterecek şekilde nesneyi ifade edebiliriz. Daha basit bir yaklaşım kullanarak resmi, her bir pikselin parlaklığına karşılık gelen bir değer sırası ile ifade edebiliriz. Önemli olan, ifade tekniğinin, çözülmekte olan problemin tipi göz önüne alındıktan sonra seçilmesidir.

4.2.2 Problemin İfade Edilmesi ve Çözülmesi

Problem çözme spesifik bir hedefe ulaşma işidir. Problemlerin çözülebilmesi için ilk olarak tartışılabilecek ve çözülebilecek şekilde ifade edilebilmeleri gerekir. Problemlerin ifadesi için kullanılan iki tip şema şunlardır.

1. **Durum-Uzay İfadesi:** Bu metotla problemi, çözüm geliştirme sırasında bulunan, tüm mümkün hallerin bir ağaç şeklinde tasarlanmış durumuyla görebiliriz. Ağaç düğümlerden

oluşur. Bu düğümler bir faaliyetten sonra sistemin aldığı durumları ifade eder. Faaliyetler de düğümleri birleştiren yaylarla gösterilir. Bu Gezgin Satıcı problemi ile açıklanabilir. Gezgin Satıcı A,B,C,D olarak 4 şehri gezmelidir. Geziye A şehrinde başlayıp, A şehrinde bitirmek istemektedir. (Şekil 18.2)



Şekil 18.2: Gezgin Satıcı problemi ağacı

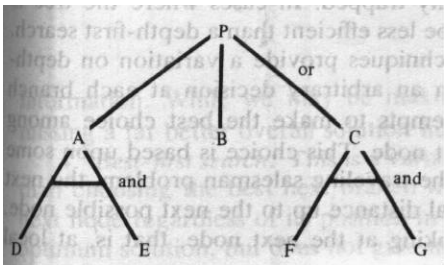
	A	B	C	D
A	0	3	4	5
B	3	0	7	6
C	4	7	0	2
D	5	6	2	0

Şehirler arasındaki mesafe

Durum-Uzay ifadesine bakarak, çözümün en küçük yay uzunluğuna sahip dalın seçilmesi olduğu görülmektedir. Bu tip problem çözümüne "forward reasoning" denilmektedir. Çünkü ileriye doğru, tüm olası yollar boyunca gidilerek sonuca ulaşılmıştır.

2. **Problem-İndirgeme İfadesi:** Bu olayda da "backward reasoning"'e örnek verilecektir. Bu ifade şeklinde, hedefi ana veri parçası olarak ele alıp, **ilkel problemler** kümesi bulunana kadar problem indirgenir. İlkel problemler, elimizde verileri bulunan daha basit problemlerdir. Bu basitleştirme, problemi, tümü çözülmesi gereken küçük problem kümelerine veya çözülebilecek alternatif problemlere parçalayarak gerçekleştirilir. Şema "VE-VEYA" grafik gösterimi ile ifade edilmiştir. "VE-VEYA" grafiğinde yatay çizgilerle birleştirilmiş yaylar "VE'lenmiş", yatay çizgilerle birleştirilmemiş yaylarda "VEYA'lanmıştır". (Şekil 18.3).

a) P, A **veya** B **veya** C çözülerek çözülebilir.
b) A, D **ve** E çözülerek çözülebilir.
c) B doğrudan çözülebilir (B bir İLKELdir).
d) C, F **ve** G çözülerek çözülebilir.
e) D, E, F ve G İLKELlerdir.



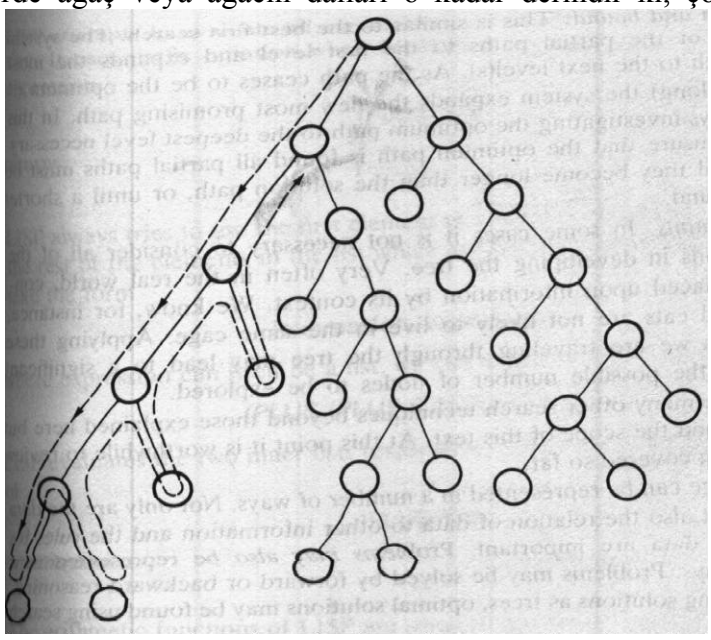
Şekil 18.3: VE-VEYA grafiği

Bu durumda P'yi çözmenin en kolay yolu B'yi çözmektir.

4.2.3 Problem Çözümünde Arama Teknikleri

Problemi veya bilgiyi, en iyi ifade tekniği ile ele aldıktan sonra hedefe ulaşmak için yapılması gereken tek iş çözümü aramaktır. Çoğu durumda gerekli olan arama uygun bir sürede bitirilemeyecek kadar büyüktür. Örnek olarak satrancı ele alalım. Belirli bir başlangıç durumuna karşılık hedef karşı oyuncunun şahını ele geçirmektir. Bütün hareket kombinasyonlarını durum-uzay ifadesi ile geliştirerek, yolları izleyip en iyi kazanan sonucu bulmak mümkündür. Ama maalesef bu hareketlerin toplam kombinasyonu $10^{20'}$dir. Bu "kombinasyonel patlamadan" dolayı arama işini en aza indirmeye çalışan sistemler geliştirilmiştir. Bu, aramanın elde bulunan tek teknik olduğu anlamına gelmemelidir. Çeşitli durumlar için geliştirilen arama teknikleri aşağıda sıralanmıştır.

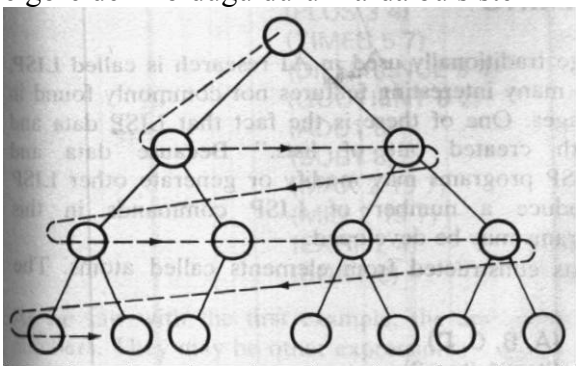
1. **Derinlik Öncelikli Arama (Depth-First Search):** Adından da anlaşılacağı gibi bu teknik ilk olarak ağaç ağında sonucu bulmak için en derine kadar (en uç düğüme kadar) arar. Amaç "S" ile belirtilen ve de çözüm anlamına gelen düğümü bulmaktır (Şekil 18.4). Eğer o düğüm istenilen sonuç değilse o zaman ilk uygun dallanma noktasını bulana kadar geri döner ve buradan tekrar en derine iner. Verilen örnekte sistem, sonunda çözüm düğümünü bulacaktır. Ama bazı sistemlerde ağaç veya ağacın dalları o kadar derindir ki, çözüm hiçbir zaman bulunamamaktadır.



4. Şekil 18.4: Derinlik Öncelikli Arama
5.

2. **Genişlik Öncelikli Arama (Breadth-First Search):** Sistem ilk olarak aynı seviyede bulunan düğümleri değerlendirir ve ondan bir alt seviyeye geçer (Şekil 18.5). Bu sistem ilkine

göre daha tutucudur ki bu da sistemin kolayca tuzağa düşmemesini sağlar. Ağacın dallarının birbirine göre derin olduğu durumlarda bu sistem ilkine göre daha verimsiz olabilir.



Şekil 18.5: Genişlik Öncelikli Arama

3. **Tepe Tırmanma (Hill Climbing):** Bu teknik **derinlik öncelikli arama** için bir varyasyon sağlar. Tepe tırmanma algoritması, her bir dallanma noktasından keyfi olarak hareket etmek yerine, üzerinde bulunulan düğümden dallanan olası tüm dallar arasından en iyisini seçmeye çalışır. Bu seçim bazı tekniklerine dayanır. Mesela, Gezgin Satıcı probleminde bir sonraki düğüm, bir sonraki düğümlere kadar olan uzaklıkların toplamına bağlı olarak seçilebilir. Buradaki risk ise, yerel bilgiye dayanarak yalnızca bir sonraki düğümlere bakmamızdır. En iyi yerel seçimleri yaparken, daha önceden kötü bir ilişkiye sahip ama toplamda en iyi olan bir sonucu kaçırabiliriz.

4. **En İyi Öncelikli Arama**: Bu Tepe Tırmanma tekniğinin bir varyasyonudur. Bu teknikte sistem, bir düğüm üzerindeki en iyi dalı seçmek yerine, düğümün yerinden bağımsız olarak bir sonraki en iyi düğümü seçer. Bu genelde optimum çözümü verir ama garanti değildir.

5. **Dal ve Sınır:** Bu En İyi Öncelikli Aramaya benzer. Sistem tüm kısmi yolları ilk aşamada değerlendirir ve de en iyi çözümü veren yolu bir sonraki aşamalara doğru genişletir. Yol optimum olarak bittikçe (çok uzunlaşır), sistem yeni bir en iyi yolu genişletir. Böylece sistem her zaman optimum yolu, derinlere doğru gerektiği kadar inceler. Optimum yolun bulunduğundan emin olmak için diğer tüm kısmi yollar, çözüm yolundan daha uzun oluncaya kadar veya daha kısa bir çözüm yolu bulunana kadar genişletilmelidir.

6. **Kısıtlar:** Bazı durumlarda ağacı oluşturan tüm olası seçeneklerin değerlendirilmesi gerekmeyebilir. Gerçek dünya da, bilgilerin içeriğine yerleştirilen bazı kısıtlar vardır. Bu kısıtları uygulayarak, ağaçta yaptığımız aramalarımızda ele almamız gereken düğümlerin sayısı önemli ölçüde azaltılabilir.

Sonuç olarak bilgi çok değişik biçimlerde ifade edilebilir. Yalnızca veriler değil, verilerin diğer bilgilerle ilişkisi ve de verilerin yönetilebilmesine ilişkin kurallar da çok önemlidir. Problemler de çok değişik şekillerde ifade edilebilir. Problemler "forward ve backward reasoning" yollarıyla çözülebilir ve de çözümleri ağaç şeklinde ifade edip, arama teknikleri kullanarak optimum çözümleri elde edebiliriz.

4.3 LISP PROOGRAMLAMA DİLİ

Yapay zeka araştırmalarında genel olarak LISP programlama dili kullanılır. LISP'te diğer programlama dillerinde bulunmayan ilginç özellikler vardır. LISP verilerinin ve LISP programlarının listelerden yaratılması, bu özelliklerden biridir. Veriler ve programlar aynı olduğundan, LISP programları diğer LISP programlarını değiştirebilir veya oluşturabilir.

LISP programları **atom** denilen elementlerden oluşan listelerdir. Aşağıdakiler listelerdir:

(A, B, C, D)
(Item1, Item2)

((Item1 A)(Item2 B))

A, B, Item1 ve **NIL** ise birer atomdur.

LISP bütün ifadeleri değerlendirip, bunun sonucunda değeri geri döndürmeye çalışır. İfadeleri değerlendirirken önek notasyonunu kullanır. **(PLUS 3 5)** ifadesi geriye **8** değerini döndürür. LISP genelde listenin ilk elemanını, listedeki diğer elemanları (argüman) değerlendirmek için, **komut** olarak kullanır. Bu yüzden listeler her zaman (ifade, ifade) formunda yazılırlar. Burada ifadelerde birer liste (PLUS (PLUS 2 3)(PLUS 4 5)) olabilirler. LISP içteki iki listenin (ifade) sonucunu (PLUS 5 9) olarak bulur. Bunun sonucunu da 14 olarak döndürür. LISP'in aritmetik fonksiyonları diğer fonksiyonların yanında çok kolaydır.

(PLUS 3 4)	7
(TIMES 5 7)	35
(DIFFERENCES 9 4)	5
(QUOTIENT 8 2)	4
(ADD 1 8)	9
(SUB 1 8)	7
(MAX 2 19 7)	19
(MIN 2 19 7)	2
(EXPT 2 4)	16
(SQRT 9)	3

İlk örnekte görüldüğü gibi argümanların sayı olmasına gerek yok. Bunlar başka ifadeler de olabilir.

LISP'in ilginç özelliği listeleri yönetmesiydi. Bu amaç için kullanılan iki fonksiyon CAR ve CDR'dır. CAR listedeki ilk elemanı, CDR ise bunun dışındaki her şeyi geri döndürür.

Örnek:

(CAR '(A B C))	A
(CAR '((A B)(C D)))	(A B)
(CDR '(A B C))	(B C)
(CDR '((A B)(C D)))	((C D))

CAR geriye atom veya liste döndürebilirken, CDR ise geriye yalnızca liste döndürebilir. Eğer CDR bir atom üzerinde işletilmeye çalışılırsa, hata döndürür. CDR, bir tek elemanlı liste üzerinde işletilmeye çalışıldığında da, geriye NIL döndürür.

Daha önceden açıklandığı gibi, LISP verilen ifadeleri değerlendirebilmek için her listedeki ilk elemanı fonksiyon olarak kullanır. Yukarıdaki örneklerde kullanılan tırnak işaretleri bu değerlendirmeyi engeller; bu şekilde LISP liste ile fonksiyon arasındaki farkı gösterebilir.

Bir ifadede hem CAR hem de CDR bulunabilir. (CAR(CDR(CDR(CDR'(A B C D))))) ifadesi geriye D'yi döndürür.

Eğer ifadeleri birbirinden ayırabiliyorsak, bunu onları tekrar bir araya koymanın bir yolu olarak da kullanabiliriz. Bu amaç için kullanılan 3 fonksiyon vardır: APPEND, LIST ve CONS.

APPEND, kendi listelerinin elemanlarını bir arada çalıştırır. Örnek olarak, (APPEND '(A B)'(C D)) ifadesi, (A B C D) sonucunu döndürür.

LIST, argümanlardan yeni listeler oluşturur. Örnek olarak, (LIST '(A B)'(C D)) ifadesi, ((A B)(C D) sonucunu döndürür.

CONS, ilk argümanı, ikinci argümanın listesindeki yeni ilk eleman olarak araya sıkıştırır. Örnek olarak, (CONS '(A B)'(C D)) ifadesi, ((A B) C D) sonucunu gönderir.

CONS, CAR ve CDR'nin bozduğunu tekrar yapabilir. Örnek olarak, (CONS (CAR'(A B C D))(CDR'(A B C D))) ifadesi, (A B C D) sonucunu gönderir.

SETQ, bir ifadeyi isimlendirmek için kullanılır. Örnek olarak, (SETQ VW '(A TYPE OF CAR)) ifadesinde "A TYPE OF CAR" değeri VW argumanına atanır. Artık LISP, VW argumanını her gördüğünde geriye A TYPE OF CAR değerini gönderecektir.

(CAR VW) A
(CDR VW) TYPE OF CAR

DEFUN (DEFine FUNction) komutu, yeni bir fonksiyon tanımlamak için kullanılır. DEFUN fonksiyonu aşağıdaki biçimde kullanır.

(DEFUN <fonksiyonun adı>
(<parametre 1>,<parametre 2>.....<parametre n>)
(<fonksiyonun tanımı>))

Feet ve inç birimindeki yükseklik değerini yalnızca ince dönüştüren fonksiyon şu şekilde olacaktır:

```
(DEFUN FT-TO-IN(FT IN)
                (PLUS (TIMES FT 12) IN))
```

Fonksiyona 5 ve 10 değerleri girildiğinde, (FT-TO-IN (5 10)), 70 sonucu alınacaktır.

Daha önceden belirtildiği gibi; belirli operasyonların başarılarını değerlendirebilmek için kararlar alınmalıdır. LISP bu iş için **yüklemlere** (predicates) sahiptir. Yüklemler yalnızca olası iki tepki verebilen fonksiyonlardır: doğru için T ve NIL. Yüklemlere örnek olarak:

(LESSP 2 3) T
(LESSP 2 3) NIL
(GREATERP 2 3) NIL
(GREATERP 2 3) T, verilebilir.

Diğer yüklemler:

AND: Listedeki tüm elemanlar nonNIL ise, geriye T döndürür.
OR: Listede nonNIL olan bir eleman varsa, geriye T döndürür.
NOT: Argümanı NIL ise, geriye T döndürür.
ZEROP: Argümanı 0 ise, geriye T döndürür.
EQUAL: Birbirine eşit iki argümanı varsa, geriye T döndürür.

Yüklemler **COND** fonksiyonu ile birleşerek, sık sık test olarak kullanılır. COND aşağıda gösterilen biçimde kullanılır.

(COND (<test 1>.....<sonuç1>)
(<test2>.....<sonuç2>)

..
(<testn>.....<sonuçn>)

Bir COND komutu hüküm **(clause)** denilen, bir liste serisinden oluşur. LISP her hükmün ilk elemanını bir nonNIL sonucu (bir T olmasına gerek yoktur) bulana kadar değerlendirir. Bir nonNIL hükmü bulduğunda da, o listeyi son ifadeye kadar değerlendirir ve bulduğu sonucu geriye döndürür.

LISP'te veriler listeler halinde gösterilir ve bu listeleri kullanabilmek için bir çok komut vardır. Özellik listesinin yapısını tanımlamak için DEFSTRUCT komutu bulunur. Kullanılış biçimi:

(DEFSTRUCT(<ad>)
(<özellik adı 1><varsayılan özellik>)
(<özellik adı 2><varsayılan özellik>)
...
(<özellik adı n><varsayılan özellik>))

DEFSTRUCT komutunun değerlendirilmesi yalnızca özellik listesinin yapısını kurmakla kalmaz aynı zamanda **MAKE-name** fonksiyonunu da oluşturur. MAKE-name, özellik listesine değer atamada kullanılır. DEFSTRUCT ayrıca "özellik adı n" olarak adlandırılan, "seçici (selector)" metotlar oluşturur.

ÖRNEK:
DEFSTRUCT yapısını kullanarak köpekleri tanıtan bir özellik listesi oluşturalım. Köpeklerin dört ayaklı olduğunu kabul edelim; bu çok iyi bir varsayılan özelliktir. Ayrıca köpeklerin renklerini de bilmek istediğimizi düşünelim. Ama bu değişken olduğunda, varsayılan özellik NIL olacaktır. Son olarak ta cinslerini, sportif ve evcil olarak tanımlayalım.

(DEFSTRUCT (DOG)
(COLOR NIL)
(NUMB-OF-FEET 4)
(TYPE NIL))

Oluşturduğumuz yapı için SETQ komutunu da kullanarak, MAKE-DOG fonksiyonunu kullanıp bir özellik listesi oluşturabiliriz.

(SETQ BEAGLE (MAKE-DOG))

Bu ifade ile BEAGLE adlı, bir özellik listesi oluşturulur. DOG-COLOR, DOG-NUMB-OF-FEET ve DOG-TYPE seçici metotlarını kullanarak, BEAGLE'ın özelliklerini inceleyebiliriz. Bu durumda bu üç metot ta, NIL, 4 ve NIL varsayılan değerlerini göndereceklerdir. SETQ komutunu ve de alan tanımlayıcılarını kullanarak, özelliklere yeni değerler atayabiliriz.

(SETQ BEAGLE (MAKE-DOG:COLOR 'BROWN:TYPE 'SPORTING))

Seçici metotları değerlendirirsek:

(DOG-COLOR BEAGLE)
(DOG-TYPE BEAGLE)

sonuç olarak:

BROWN
SPORTING, döndürülür.

Özellik listesindeki belirli bir alana değer atamak için SETF fonksiyonu kullanılır.

(SETF(<özellik><adı>)(yeni değer))

Örnek olarak, (SETF(DOG-COLOR BEAGLE) 'PURPLE)) yazarak, Beagle'ı mor köpek olarak değiştirebiliriz.

4.4 YAPAY ZEKA VE ROBOT BİLİMİ

Yapay zeka konusundaki çalışmalar gelecekte, robot sistemlerinin operasyonlarına ve görme gibi duyu fonksiyonlarına ışık tutması açısından çok önemlidir.

Görüş sistemlerinde verilerin ifade edilişi sorun olmaktadır. Sistem belleğinde parçaların resimlerinin depolanması yerine, nesnenin, çevre, alan, delik sayıları vb. gibi, belirli özellikleri depolanmaktadır. Sistem üç boyutlu kalabalık ortamlarla çalışıp gittikçe karmaşıklaşınca, görüş sistemi çok daha büyük bir zekaya ihtiyaç duyacaktır. Görüş sistemlerinin, düzensiz olarak toplandıklarında dahi, bloklar gibi özel tipteki nesneleri tanıyabilmelerini sağlayan teknikler geliştirilmiştir. Bu sistemler, işlem hızını arttırmak için, ağırlıklı olarak *aramayı en küçükleme yöntemlerine* dayanırlar.

Diğer bir çalışma alanı da görev seviyeli programlamadır. Robota gerekli olan hareketleri adım adım yaptırmak yerine, "büyük kırmızı parçayı al" gibi ifadeler oluşturmak mümkün olacaktır. 1971'de geliştirilen, SHRDLU adlı bir program operatörün robotla konuşmasını sağlıyordu. SHRDLU bir görev planlayıp, buna tepki verebilme özelliğine sahipti. Mesela, eğer kırmızı parça yeşil parçanın altındaysa, kırmızı parçayı almadan önce, yeşil parçayı alıp onu yere koyacaktır.

Gerçekte fabrikalar sınırlı ortamlardır. Bu da, robot zekasını pratikleştirme konusundaki problemleri sınırlamaktadır.

4.5 FABRİKADA LISP

Bu bölümde, fabrikadaki basit bir işe rehberlik edecek bir LISP programı geliştirilecektir. Robotun yapması gereken iş, bir küçük dişli, bir büyük dişli ve de iki şaftlı bir levhadan oluşan dişli kutusunun montajıdır. Bir oluktan çalışma alanına boşaltılan parçalar robota, bir kamera ile tanıtılmaktadır. Bu besleme tekniğinden dolayı parçalar düzensiz bir şekilde hatta üst üste duruyor olabilir. Problemi basitleştirmek için bazı kabuller yapacağız:

1. Kamera parçaları ve yerlerini tanıyabilir.
2. Kamera bu üç parçayı her zaman bulabilir.
3. Robot bu üç parçayı, ne şekilde dururlarsa dursunlar, her şekilde kavrayabilir.
4. Parçalar arasındaki ilişkiler görüş sistemi ile tanımlanır; yani görüş sistemi eğer bir parça diğerinin üzerindeyse bu belirtir.

Şimdi işi daha detaylı olarak tanımlamalıyız.

1. Robot levhayı sabit bir montaj donanımına yerleştirecek.
2. Büyük dişliyi yerine yerleştirecek (GEAR 1 TO PIN 1)
3. Küçük levhayı yerine yerleştirecek (GEAR 2 TO PIN 2)

Eğer herhangi bir parça diğerinin yolu üzerindeyse, robot ilk olarak engel olan bu parçayı ortadan kaldırmalıdır. Bu açıklamalardan sonra işi tanımlarsak:

```
(PLACE 'PLATE FIXTURE)
(PLACE 'GEAR1 PIN1)
(PLACE 'GEAR1 PIN1)
```

Konumları da şu şekilde tanımlayabiliriz.

```
(SETQ FIXTURE xyz
      PIN1 abc
      PIN2 lmn)
```

Burada xyz, abc ve lmn gerçek pozisyon değerleridir. Şimdi yapılması gereken, PLACE fonksiyonunun tanımlanmasıdır.

```
(DEFUN PLACE (PART LOCATION)
      (GRAB (PART))
      (PUTON (PART LOCATION)))
```

Burada PLACE fonksiyonunu iki metotla tanımladık. İşi ilkel fonksiyonlar şeklinde açıklayana kadar yeni metotlar tanımlaya devam edeceğiz. Robotun operasyonunu kolaylaştırmak için bazı LISP ilkelleri ekleyelim:

OPEN	Tutucuyu (gripper) açar.
CLOSE	Tutucuyu kapar.
MOVE (LOCATION)	Kolu belirtilen konuma götürür.

Sistem parçayı kavrayabilmek için, ilk olarak parçanın başka bir parça tarafından çevrelenmiş olup olmadığını kontrol etmelidir. Ayrıca sistem parçanın konumunu bilmelidir. Eğer bu özellikleri içeren bir özellik listesi oluşturursak, istenilen değerleri bulabiliriz.

```
(DEFSTRUCT (OBJECT)
            (POSITION NIL)
            (IS-UNDER NIL))
```

Eğer görüş sistemi parçanın konumunu ve de başka bir parça tarafından kavranmış olup olmadığını belirleyebilecekse:

```
(SETQ GEAR1 (MAKE-OBJECT :POSITION P1 :IS-UNDER U1)
      GEAR2 (MAKE-OBJECT :POSITION P2 :IS UNDER U2)
      GEAR3 (MAKE-OBJECT :POSITION P3 :IS UNDER U3))
```

Burada kullanılan P1, P2, P3, U1, U2 ve U3 değerleri görüş sistemi tarafından tanımlanır.

GRAB adlı bir diğer fonksiyonda engel olup olmadığını kontrol eder ve bu engeli ortadan kaldırıp parçayı kavrar.

```
(DEFUN GRAB (PART)
      (COND (OBJECT-IS-UNDER PART)(CLEAROFF PART))
      (GRASP PART))
```

Eğer OBJECT-IS-UNDER geriye değer döndürürse, ilk önce CLEAROFF daha sonra da GRASP komutu gerçekleştirilir. Eğer OBJECT-IS-UNDER geriye bir değer döndürmezse o zaman da doğrudan GRASP komutu gerçekleştirilir.

```
(DEFUN GARSP (PART)
      (MOVE (OBJECT-POSITION PART))
      (CLOSE)
      (SETQ GRIPPER PART))
```

GRASP komutu parçayı belirlenen konuma yerleştirir, tutucuyu kapatır ve tutucuya parçanın adını verir.

PUTON fonksiyonu, parçayı istenilen konuma getirmek için kullanılır.

```
(DEFUN PUTON (PART LOCATION)
      (MOVE LOCATION)
      (UNGRASP PART))
```

UNGRASP fonksiyonu, tutucuyu açarak parçayı bırakmak için kullanılır.

```
(DEFUN UNGRASP (PART)
        (OPEN)
        (SETQ GRIPPER NIL))
```

En son olarak ta CLEAROFF fonksiyonunu tanımlarsak:

```
(DEFUN UNGRASP (PART)
        (SETQ TROUBLE (OBJECT-IS-UNDER PART))
        (SETQ CLEARSPOT FREESPACE)
        (GRASP TROUBLE)
        (PUTON TROUBLE CLEARSPOT)
        (SETF (OBJECT-POSITION TROUBLE) CLEARSPOT)
        (SETF (OBJECT-IS-UNDER PART) NIL))
```

Buradaki FREESPACE fonksiyonu, görüş sistemine bir boş alan değeri sorar. Sistem bu boş alanı TROUBLE adlı engeli koymak için kullanır. Ayrıca engelin eski konumu ile ilgili değerleri silerek yeni konumunu değerlendirir. Daha sonra da engel olan parçayı yerine koyarak, engeli ve engel ile ilgili değerleri ortadan kaldırmış olur.

Bu örnekte bir fabrikada karşılaşılabilecek büyük bir problem, nesnelerin birbirleri ile ilişkisi ele alınarak, küçük, çözülebilir işlere ayrıştırılarak çözülmüştür.

5. KAYNAKÇA

1. Groover M.P., Weiss M.M, Nagel R.N., Odrey N.G., 1986, Industrial Robotics Technology, Programming and Applications, McGraw Hill
2. Snyder W.E., Industrial Robots Computer Interfacing and Control
3. Asfahl, C.R., Robots and Manufacturing Automation
4. Russell A.R., Robot Tactile Sensing
5. Critchlow A.J., 1985, Introduction to Robotics, Macmillan Publishing Company
6. Tübitak Bilim - Teknik Dergisi Haziran 1992 sayısı, "Teknoloji Harikası Robot Ellerin Tasarımı"
7. Eyler M.A., Haziran 1986, Robotlar ve Endüstriyel Otomasyon Semineri-Endüstride Robot Uygulamaları
8. Kucur, A.F., Yüksek Lisans Tezi
9. Dilaver K.F., Robot Kollarının Adaptif Kontrolü, Yüksek Lisans Tezi
10. Başpınar C., Robot Kollarının Gürbüz Kontrolü, Yüksek Lisans Tezi