LAPORAN TUGAS BESAR I
IF4070

Pengembangan Ontologi dan Sistem Berbasis Pengetahuan untuk Domain Dota 2

oleh

SA'AD ABDUL HAKIM - 13522092

RAYHAN FADHLAN AZKA - 13522095

RAYENDRA ALTHAF TARAKA NOOR - 13522107

PROGRAM STUDI TEKNIK INFORMATIKA

SEKOLAH TEKNIK ELEKTRO & INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

November 2025

Daftar Isi

1. Domain

1.1. Penjelasan Domain

Domain yang kami pilih untuk proyek ini adalah Dota 2, sebuah permainan video bergenre *Multiplayer Online Battle Arena* (MOBA) yang sangat populer. Dalam permainan ini, dua tim yang masing-masing terdiri dari lima pemain bertarung satu sama lain, di mana setiap pemain mengontrol satu karakter yang disebut "Hero". Setiap Hero memiliki atribut, peran (*role*), dan serangkaian kemampuan (*abilities*) yang unik. Tujuan utama permainan adalah untuk menghancurkan struktur utama tim lawan.

Domain ini sangat kaya akan entitas, properti, dan relasi yang terstruktur, menjadikannya kandidat yang sangat baik untuk dimodelkan dalam sebuah ontologi. Terdapat lebih dari 120 Hero, ratusan ability, dan item yang memiliki atribut dan interaksi yang jelas.

1.2. Batasan Domain

Meskipun domain Dota 2 sangat luas, kami membatasi cakupan ontologi dan sistem kami pada aspek-aspek statis dari permainan, yaitu:

- **Hero:** Karakter yang dapat dimainkan, beserta atribut utamanya (Strength, Agility, Intelligence, Universal), tipe serangan (Melee/Ranged), dan peran dalam tim (Carry, Support, dll.). Karakter yang tidak dapat dimainkan juga dituliskan, namun hanya terdapat pada ontologi dan tidak memiliki atribut, hanya keberadaannya.
- **Kemampuan (Abilities):** Kemampuan unik yang dimiliki oleh setiap Hero, termasuk tipe perilaku (*behavior*) dan tipe serangan (*damage type*).
- **Item:** Perlengkapan yang dapat dibeli dalam permainan, namun dalam ontologi ini, kami tidak memodelkan efek aktif/pasif item secara mendalam, melainkan hanya keberadaannya.
- **Relasi Antar Entitas:** Kami fokus pada relasi kepemilikan, seperti hero - hasAbility -> ability dan hero - hasRole -> role.

Kami tidak memodelkan aspek-aspek dinamis seperti:

- Status permainan (posisi hero, level saat ini, item yang dimiliki).
- Strategi permainan yang kompleks.
- Interaksi spesifik antar kemampuan (misalnya, *counter*).
- Statistik pemain atau histori pertandingan.

1.3. Kecocokan Domain

Domain Dota 2 sangat cocok untuk proyek ini karena beberapa alasan:

1. **Struktur Jelas:** Terdapat hierarki dan klasifikasi yang jelas. Contohnya, kelas Hero dapat dibagi lagi menjadi StrengthHero, AgilityHero, dll. Hal ini memungkinkan penerapan konsep subclass dalam Logika Deskripsi.

2. **Banyak Fitur Logika Deskripsi:** Domain ini kaya akan properti dan relasi yang dapat dimodelkan. Contohnya, properti objek seperti hasPrimaryAttribute dan hasAbility, serta properti data seperti attackRange dan movementSpeed.
3. **Tidak Terlalu Luas atau Sempit:** Dengan batasan yang kami tetapkan, domain ini cukup kaya untuk dieksplorasi tanpa menjadi terlalu kompleks untuk dimodelkan dalam lingkup waktu proyek.
4. **Sumber Data Terstruktur:** Data mengenai Dota 2 tersedia secara luas melalui API publik (seperti OpenDota) dan data feed dari situs resminya, yang dapat kami proses secara otomatis dengan skrip python.

2. Ontologi

2.1. Sumber Pengetahuan

Sumber pengetahuan utama kami berasal dari data feed resmi Dota 2, API publik OpenDota (sumber : https://github.com/odota/dotaconstants/tree/master/build). Data dari OpenDota bersifat JSON yang langsung bisa diproses, untuk data mengenai struktur bangunan (tower dan fountain) dan data creep kami mendapat data dari feed resmi Dota 2. Format JSON dari OpenDota :

- heroes.json: Berisi detail statistik dan atribut dasar setiap Hero.
- abilities.json: Berisi deskripsi dan properti dari semua kemampuan dalam permainan.
- items.json: Berisi informasi mengenai item.
- hero_abilities.json: Memetakan Hero dengan ability, talent, dan facet spesifik mereka.

2.2. Gambaran Umum Ontologi (TBox)

TBox (Terminological Box) merupakan komponen dari basis pengetahuan yang mendefinisikan terminologi atau kosakata dari domain yang dimodelkan. Dalam proyek kami, TBox berfungsi sebagai ske ma konseptual yang mendefinisikan semua kelas (konsep) dan properti (peran) yang relevan dalam domain Dota 2. Struktur ini kami bangun secara terprogram menggunakan skrip Python dengan bantuan pustaka owlready2, yang memungkinkan kami untuk mendefinisikan hierarki kelas dan relasi secara formal sebelum mempopulasinya dengan data individu (ABox).

2.2.1. Hierarki Kelas

Hierarki kelas dirancang secara top-down, dimulai dari konsep yang paling umum hingga konsep yang lebih spesifik. Struktur ini secara akurat merepresentasikan taksonomi entitas dalam dunia Dota 2.

- **Dota2Entity** (Kelas dasar untuk semua entitas)
    - **Agent** (Entitas yang dapat bertindak)
        - **Hero** (Karakter yang dapat dimainkan)
            - StrengthHero
            - AgilityHero
            - IntelligenceHero

        - UniversalHero
    - **NonHeroUnit** (Unit selain Hero)
        - **Creep**
            - LaneCreep
            - NeutralCreep
        - SummonedUnit
        - Roshan
- **GameItem** (Item dalam permainan)
    - **Item**
        - ComponentItem
        - CraftedItem
        - ConsumableItem
        - NeutralItem
- **AbilityConcept** (Konsep terkait kemampuan)
    - **Ability** (Kemampuan aktif atau pasif)
        - BasicAbility
        - UltimateAbility
    - Talent
    - Facet
- **GameMechanic** (Konsep mekanik permainan abstrak)
    - Role
    - Attribute
        - PrimaryAttribute
    - AttackType
    - DamageType
    - Behavior
- **Structure** (Bangunan dalam permainan)
    - **Building**
        - Tower
        - Barracks
        - Ancient
        - Effigy
    - **Fountain**

Sebuah aksioma penting yang kami terapkan adalah AllDisjoint yang contohnya diterapkan pada kelas StrengthHero, AgilityHero, IntelligenceHero, dan UniversalHero. Ini adalah fitur dari Logika Deskripsi yang memastikan bahwa seorang Hero tidak dapat menjadi anggota dari lebih dari satu tipe atribut utama secara bersamaan, yang sesuai dengan aturan dalam permainan Dota 2.

2.2.2. Properti

Properti digunakan untuk mendefinisikan hubungan antar individu (properti objek) dan atribut literal dari individu (properti data). Berikut adalah properti-properti utama yang kami definisikan:

Properti Objek

Properti ini mendefinisikan relasi antara dua individu dalam ontologi.

Nama Properti	Domain	Range	Deskripsi
hasAbility	Hero	AbilityConcept	Menghubungkan seorang Hero dengan kemampuan yang dimilikinya.
hasRole	Hero	Role	Menetapkan satu atau lebih peran untuk seorang Hero.
hasPrimaryAttribute	Hero	PrimaryAttribute	Menentukan atribut utama dari seorang Hero.
hasAttackType	Hero	AttackType	Menentukan tipe serangan Hero (misalnya, Melee atau Ranged).
hasDamageType	AbilityConcept	DamageType	Menentukan tipe *damage* yang dihasilkan oleh sebuah kemampuan.
hasBehavior	AbilityConcept, Item	Behavior	Mendeskripsikan perilaku sebuah *skill* (misalnya, *Unit Target*, *Passive*).
requiresComponent	CraftedItem	Item	Menunjukkan bahwa sebuah item dibuat dari item komponen lain.

Properti Data

Properti ini menghubungkan individu dengan nilai literal seperti string, angka, atau boolean.

Properti	Domain	Tipe Data	Deskripsi
displayName	Dota2Entity	string	Nama umum yang digunakan untuk entitas apa pun di dunia Dota 2.
description	Dota2Entity	string	Deskripsi teks pendek mengenai entitas.
imageURL	Dota2Entity	string	URL gambar resmi yang merepresentasikan entitas.
health	Agent, Structure	integer	Nilai HP dasar yang dimiliki unit atau struktur.
mana	Agent	integer	Nilai mana dasar unit.
healthRegen	Agent, Structure	float	Regenerasi HP per detik.
manaRegen	Agent	float	Regenerasi mana per detik.
armor	Agent, Structure	integer	Nilai armor fisik dasar.
magicResistance	Agent	integer	Persentase resistansi sihir dasar.
movementSpeed	Agent	integer	Kecepatan gerak dasar unit.
experienceBounty	NonHeroUnit, …	integer	XP yang diberikan saat unit/struktur dikalahkan.
localizedName	Hero	string	Nama hero sebagaimana muncul di dalam game.
baseMinAttackDam	Hero	integer	Nilai minimum

age			serangan dasar hero.
baseMaxAttackDamage	Hero	integer	Nilai maksimum serangan dasar hero.
attackRange	Hero, Structure	integer	Jarak serangan dasar.
attackRate	Hero, Structure	float	Interval serangan dalam detik.
strengthGain	Hero	float	Tambahan Strength per level.
agilityGain	Hero	float	Tambahan Agility per level.
intelligenceGain	Hero	float	Tambahan Intelligence per level.
dayVision	Hero	integer	Jarak pandang hero saat siang.
nightVision	Hero	integer	Jarak pandang hero saat malam.
cooldown	AbilityConcept	float	Waktu jeda sebelum ability dapat dipakai lagi.
manaCost	AbilityConcept	integer	Kebutuhan mana per penggunaan ability.
requiredLevel	Talent	integer	Level hero minimal untuk membuka talent.
facetTitle	Facet	string	Judul singkat facet.
facetDescription	Facet	string	Penjelasan detail facet.
facetColor	Facet	string	Kode/label warna facet.
cost	Item	integer	Harga item dalam

			emas.
lore	Item	string	Latar cerita singkat item.
notes	Item	string	Catatan atau tips terkait item.
tier	NeutralItem	integer	Tingkatan drop neutral item.
goldBounty	Structure	integer	Gold yang diberikan saat struktur hancur.
structureTier	Tower	integer	Tier menara (1–4).
trueSightRadius	Tower, Fountain	integer	Radius True Sight yang dimiliki.
goldBountyMin	Creep	integer	Bounty emas minimum creep.
goldBountyMax	Creep	integer	Bounty emas maksimum creep.
spawnLocation	NonHeroUnit	string	Lokasi spawn unit non-hero.

Struktur TBox yang terdiri dari hierarki kelas, properti objek, dan properti data ini membentuk fondasi logis yang kuat untuk merepresentasikan pengetahuan domain Dota 2. Selanjutnya, ABox akan mengisi struktur ini dengan fakta-fakta spesifik tentang mereka.

2.3. Proses Pembangunan Ontologi (ABox)

ABox (asersi) atau populasi individu dalam ontologi kami dilakukan secara terprogram menggunakan skrip Python. Skrip ini melakukan langkah-langkah berikut:

1. **Membaca Data JSON:** Memuat semua file sumber data (heroes.json, abilities.json, dll.) ke dalam memori.
2. **Mendefinisikan Struktur TBox:** Menggunakan owlready2, skrip mendefinisikan semua kelas dan properti yang telah dirancang.
3. **Membuat Individu:** Melakukan iterasi pada setiap entitas dalam file JSON (setiap Hero, setiap Ability, dll.) dan membuat individu baru dalam ontologi.
4. **Menambahkan Fakta (Assertions):** Untuk setiap individu, skrip menambahkan fakta-fakta berdasarkan datanya. Sebagai contoh, untuk Hero "Axe":
    - Membuat individu axe dari kelas StrengthHero.
    - Menambahkan fakta hasPrimaryAttribute(axe, strength).

    - Menambahkan fakta hasRole(axe, carry), hasRole(axe, initiator), hasRole(axe, disabler).
    - Menambahkan fakta hasAbility(axe, berserkers_call), hasAbility(axe, battle_hunger), dst.
5. **Menyimpan Ontologi:** Setelah semua individu dan fakta ditambahkan, ontologi disimpan ke dalam file dota2_ontology.owl dengan format RDF/XML, yang dapat dibuka dan diperiksa menggunakan Protégé serta bisa dikonversi ke bentuk .rdf.

3. Sistem Berbasis Pengetahuan

Sistem Berbasis Pengetahuan (KBS) kami dikembangkan menggunakan Prolog untuk melakukan penalaran simbolik terhadap pengetahuan yang telah dimodelkan dalam ontologi Dota 2. Sistem ini dirancang untuk melakukan klasifikasi hero berdasarkan serangkaian aturan yang merefleksikan arketipe dan peran strategis yang umum dikenal dalam permainan.

3.1. Arsitektur Sistem

Arsitektur KBS kami di Prolog dirancang secara modular dengan tiga lapisan logika yang berbeda, yang semuanya diintegrasikan melalui file utama tubes1_main.pl.

1. **Lapisan ABox (abox_dota2.pl):** Ini adalah lapisan dasar yang berisi fakta-fakta mentah. File ini dihasilkan secara otomatis oleh skrip python dan merupakan representasi langsung dari ABox ontologi OWL. Lapisan ini berisi asersi fundamental seperti hero(axe), primary_attribute(axe, strength), dan has_role(axe, initiator).
2. **Lapisan TBox (tbox_dota2.pl):** Lapisan ini berfungsi sebagai jembatan antara fakta mentah dan aturan tingkat tinggi. Ia berisi aturan-aturan sederhana yang mengabstraksikan konsep-konsep terminologis (TBox) dari ontologi. Sebagai contoh, alih-alih memeriksa primary_attribute(Hero, strength) secara langsung, kita dapat menggunakan predikat yang lebih mudah dibaca is_strength_hero(Hero). Lapisan ini membuat aturan KBS menjadi lebih deklaratif dan mudah dipahami. Berikut list aturan yang tersedia pada lapisan TBox:

Aturan (Predicate)	**Deskripsi**
Berdasarkan Atribut Utama	
is_agility_hero(Hero)	Mendefinisikan hero sebagai 'Agility Hero' jika atribut utamanya adalah agility.

is_intelligence_hero(Hero)	Mendefinisikan hero sebagai 'Intelligence Hero' jika atribut utamanya adalah intelligence.
is_strength_hero(Hero)	Mendefinisikan hero sebagai 'Strength Hero' jika atribut utamanya adalah strength.
is_universal_hero(Hero)	Mendefinisikan hero sebagai 'Universal Hero' jika atribut utamanya adalah universal.
Berdasarkan Role	
is_carry(Hero)	Mengecek apakah hero memiliki role 'Carry'.
is_support(Hero)	Mengecek apakah hero memiliki role 'Support'.
is_nuker(Hero)	Mengecek apakah hero memiliki role 'Nuker'.
is_escape(Hero)	Mengecek apakah hero memiliki role 'Escape'.
is_disabler(Hero)	Mengecek apakah hero memiliki role 'Disabler'.
is_durable(Hero)	Mengecek apakah hero memiliki role 'Durable'.
is_initiator(Hero)	Mengecek apakah hero memiliki role 'Initiator'.
is_pusher(Hero)	Mengecek apakah hero memiliki role 'Pusher'.
Berdasarkan Properti Ability	

has_passive_ability(Hero)	Mengecek apakah hero memiliki setidaknya satu ability dengan tipe 'passive'.
has_aoe_ability(Hero)	Mengecek apakah hero memiliki setidaknya satu ability dengan tipe 'Area of Effect' (AoE).
has_pure_damage_ability(Hero)	Mengecek apakah hero memiliki setidaknya satu ability aktif (bukan pasif) yang menghasilkan damage tipe 'pure'.
has_channeled_ability(Hero)	Mengecek apakah hero memiliki setidaknya satu ability dengan tipe 'channeled'.
Berdasarkan Tipe Attack	
is_melee_hero(Hero)	Mengecek apakah hero memiliki tipe serangan 'melee'.
is_ranged_hero(Hero)	Mengecek apakah hero memiliki tipe serangan 'ranged'.

3. **Lapisan Aturan KBS (kbsrules_dota2.pl):** Ini adalah lapisan penalaran tingkat tinggi. Lapisan ini berisi aturan-aturan klasifikasi kompleks yang mendefinisikan arketipe hero. Aturan-aturan ini memanfaatkan predikat-predikat yang telah didefinisikan di lapisan TBox untuk membentuk logika yang lebih kompleks. Berikut list aturan yang tersedia pada lapisan aturan KBS:

Aturan (Predicate)	Deskripsi
is_hard_carry(Hero)	Mengklasifikasikan hero sebagai 'Hard Carry' jika ia memiliki role 'Carry' DAN merupakan 'Agility Hero'.

is_magic_nuker(Hero)	Mengklasifikasikan hero sebagai 'Magic Nuker' jika ia memiliki role 'Nuker' DAN memiliki setidaknya satu ability aktif (bukan pasif) yang menghasilkan damage 'magical'.
is_pure_tank(Hero)	Mengklasifikasikan hero sebagai 'Pure Tank' jika ia merupakan 'Strength Hero' DAN memiliki role 'durable'.
is_glass_cannon(Hero)	Mengklasifikasikan hero sebagai 'Glass Cannon' jika ia adalah 'Nuker' ATAU 'Carry', TETAPI TIDAK memiliki role 'durable'.
is_utility_support(Hero)	Mengklasifikasikan hero sebagai 'Utility Support' jika ia memiliki role 'Support' DAN juga 'Disabler'.
is_teamfight_controller(Hero)	Mengklasifikasikan hero sebagai 'Teamfight Controller' jika ia adalah 'Initiator' DAN memiliki setidaknya satu ability AoE.
is_ganker(Hero)	Mengklasifikasikan hero sebagai 'Ganker' jika ia adalah 'Disabler' DAN 'Nuker', TETAPI BUKAN 'Carry'.
is_elusive_escape_artist(Hero)	Mengklasifikasikan hero sebagai 'Elusive Escape Artist' jika ia memiliki role 'Escape' TETAPI TIDAK 'durable'.
is_right_click_carry(Hero)	Mengklasifikasikan hero sebagai 'Right-Click Carry' (mengandalkan serangan dasar) jika ia adalah 'Carry' DAN memiliki setidaknya satu ability pasif.
is_pure_damage_specialist(Hero)	Mengklasifikasikan hero sebagai 'Pure Damage Specialist' jika ia memiliki setidaknya satu ability aktif (bukan pasif) yang menghasilkan damage 'pure'.

is_split_pusher(Hero)	Mengklasifikasikan hero sebagai 'Annoying Split Pusher' jika ia memiliki role 'Pusher' DAN 'Escape', memungkinkannya menghancurkan tower lalu kabur.
is_spell_caster(Hero)	Mengklasifikasikan hero sebagai 'Spell Caster' jika ia adalah 'Intelligence Hero' DAN memiliki setidaknya 3 ability aktif (bukan pasif).
is_tanky_dps(Hero)	Mengklasifikasikan hero sebagai 'Tanky DPS' jika ia memiliki role 'Carry' DAN 'Durable'.
is_good_farmer(Hero)	Mengklasifikasikan hero sebagai 'Good Farmer' jika ia memiliki role 'Carry' ATAU 'Pusher' DAN memiliki setidaknya satu ability AoE, memungkinkan farming creep secara efisien.
is_strong_initiator(Hero)	Mengklasifikasikan hero sebagai 'Strong Initiator' jika ia memiliki role 'Initiator' DAN 'Disabler' DAN memiliki setidaknya satu ability AoE. Hero ini dapat memulai teamfight dengan efektif.
counters_illusions(Hero)	Mengklasifikasikan hero sebagai 'Illusion Counter' jika ia memiliki setidaknya 2 ability AoE yang menghasilkan damage physical atau magical, efektif untuk menghancurkan illusion musuh.
is_position_1(Hero)	Mengklasifikasikan hero sebagai 'Posisi 1 (Safe Lane Carry)' jika ia adalah 'Carry', BUKAN 'Support', dan merupakan 'Agility Hero' atau 'Strength Hero'.
is_position_2(Hero)	Mengklasifikasikan hero sebagai 'Posisi 2 (Mid Lane)' jika ia adalah 'Nuker' ATAU 'Carry', merupakan 'Intelligence Hero' ATAU 'Agility Hero', dan BUKAN 'Support'.

is_position_3(Hero)	Mengklasifikasikan hero sebagai 'Posisi 3 (Offlane)' jika ia 'Durable', 'Initiator', dan merupakan 'Strength Hero' atau 'Universal Hero'.
is_position_4(Hero)	Mengklasifikasikan hero sebagai 'Posisi 4 (Roaming Support)' jika ia 'Support', 'Initiator', dan 'Disabler'.
is_position_5(Hero)	Mengklasifikasikan hero sebagai 'Posisi 5 (Hard Support)' jika ia 'Support', BUKAN 'Carry', dan BUKAN 'Initiator'.

Integrasi ketiga lapisan ini dilakukan oleh tubes1_main.pl yang memuat semua file dalam urutan yang benar, memastikan bahwa aturan tingkat tinggi memiliki akses ke abstraksi dan fakta yang diperlukannya.

3.2. Fungsionalitas Sistem

Fungsionalitas utama dari KBS kami adalah untuk mengklasifikasikan seorang Hero ke dalam satu atau lebih arketipe strategis berdasarkan atribut, peran, dan kemampuan mereka. Kami telah mendefinisikan beberapa klasifikasi sebagai berikut:

- **Hard Carry:** Hero *Carry* berbasis *Agility* yang sangat bergantung pada item dan serangan fisik.
- **Magic Nuker:** Hero *Nuker* yang mengandalkan kemampuan *magic* non-pasif untuk memberikan *burst damage*.
- **Pure Tank:** Hero *Strength* yang memiliki peran *Durable*, menunjukkan kemampuannya untuk menyerap banyak kerusakan.
- **Glass Cannon:** Hero ofensif (*Carry* atau *Nuker*) yang tidak memiliki peran *Durable*, membuatnya kuat namun rapuh.
- **Utility Support:** Hero *Support* yang juga memiliki peran *Disabler*, fokus pada pengendalian musuh.
- **Teamfight Controller:** Hero *Initiator* yang memiliki setidaknya satu kemampuan *Area of Effect* (AoE) untuk mengontrol pertarungan tim.
- **Ganker:** Hero pembunuh yang lincah (*Disabler* dan *Nuker*) namun bukan *Carry*.
- **Elusive Escape Artist:** Hero dengan peran *Escape* yang lincah namun tidak *Durable*.
- **Right-Click Carry:** Hero *Carry* yang mengandalkan kemampuan pasif untuk memperkuat serangan dasarnya.
- **Pure Damage Specialist:** Hero yang mampu memberikan kerusakan tipe *Pure*, yang tidak dapat dikurangi oleh armor atau resistansi magic.

- **Annoying Split Pusher:** Hero dengan peran *Pusher* dan *Escape*, efektif untuk menghancurkan bangunan dan sulit ditangkap.
- **Spell Caster:** Hero yang berasal dari kelompok *Intelligence* dan memiliki setidaknya tiga *ability* aktif, yang membuatnya efektif dalam memberikan *magic burst*, utilitas, atau kontrol medan pertempuran melalui manajemen *spell*.
- **Tanky DPS:** Hero yang memiliki role *Carry* sekaligus *Durable*. Mereka dapat memberikan *damage* secara konsisten sambil tetap sulit dibunuh karena ketahanan yang tinggi, cocok untuk *frontline* yang tetap memukul dalam *teamfight* panjang.
- **Good Farmer:** Hero yang memiliki role *Carry* atau *Pusher* dan setidaknya satu *ability* AoE, sehingga bisa membersihkan *creep* dengan cepat. Hero tipe ini biasanya bisa mempercepat tempo item dibanding hero lain.
- **Strong Initiator:** Hero dengan role *Initiator* dan *Disabler*, serta memiliki sedikitnya satu *ability* AoE, membuat mereka sangat efektif memulai *teamfight*. Mereka dapat masuk lebih dulu, memberikan kontrol area, dan membuka peluang bagi tim untuk *follow-up*.
- **Illusion Counter:** Hero yang memiliki minimal dua *ability* AoE yang memberikan *damage physical* atau *magical*. Hero seperti ini dapat menghancurkan *illusion* musuh dengan cepat dan menjaga tim dari serangan berbasis ilusi.
- **Posisi 1 (Safe Lane Carry):** Hero yang berperan sebagai *Carry*, bukan *Support*, dan berasal dari kelompok *Agility* atau *Strength*. Hero posisi ini membutuhkan *farm* paling banyak dan menjadi sumber *damage* utama di *late game*. Mereka dilindungi oleh tim pada *early game*.
- **Posisi 2 (Mid Lane):** Hero yang merupakan *Nuker* atau *Carry*, berasal dari kelompok *Intelligence* atau *Agility*, serta bukan *Support*. Hero posisi ini biasanya bermain solo di *mid lane*, menguasai tempo permainan, dan aktif melakukan rotasi setelah mendapatkan level dan item penting lebih cepat.
- **Posisi 3 (Offlane):** Hero dengan peran *Durable* dan *Initiator*, serta berasal dari kelompok *Strength* atau *Universal*. Mereka bertugas mengganggu *carry* musuh, menjadi *frontliner*, membuka *teamfight*, dan membeli item utilitas untuk tim.
- **Posisi 4 (Roaming Support):** Hero yang memiliki role *Support*, *Initiator*, dan *Disabler*. Hero posisi ini aktif bergerak di *early game*, *ganking lane* lain, memberikan *pressure*, membuka *kill potential*, serta menyediakan *crowd control* untuk tim.
- **Posisi 5 (Hard Support):** Hero yang berperan sebagai *Support*, bukan *Carry*, dan bukan *Initiator*. Biasanya bertugas menjaga *carry* di *safe lane*, membeli *ward*, memberikan *sustain* atau perlindungan di *early game*, serta mengutamakan kebutuhan tim dibanding item pribadi.

3.3. Pemanfaatan Ontologi

Ontologi dota2_ontology.owl adalah sumber kebenaran tunggal (*single source of truth*) bagi sistem kami. Pemanfaatannya terjadi dalam dua tahap:

1. **Generasi Fakta (ABox):** Skrip python yang kami gunakan secara langsung menerjemahkan *triple* RDF dari ontologi menjadi fakta Prolog di abox_dota2.pl. Ini memastikan bahwa basis pengetahuan kami konsisten dengan model konseptual yang telah kami definisikan.
2. **Inspirasi Aturan (TBox):** Struktur TBox dalam ontologi (misalnya, hierarki kelas Hero dan properti hasRole) menginspirasi pembuatan aturan abstraksi di

tbox_dota2.pl. Dengan cara ini, struktur logis dari ontologi tetap terjaga dalam sistem Prolog, memungkinkan penulisan aturan KBS yang lebih intuitif dan modular.

Secara keseluruhan, ontologi tidak hanya berfungsi sebagai "database", tetapi juga sebagai "cetak biru" untuk struktur logis dari sistem berbasis pengetahuan kami.

4. Pembahasan

4.1. Hasil Sistem Berbasis Pengetahuan

Sistem berbasis pengetahuan kami yang dibangun di atas SWI-Prolog berhasil melakukan klasifikasi hero sesuai dengan aturan yang telah kami definisikan. Dengan memuat tubes1_main.pl, kami dapat melakukan *query* untuk mengidentifikasi hero-hero yang masuk ke dalam setiap arketipe. Berikut adalah beberapa contoh eksekusi dan hasilnya:

Contoh 1: Klasifikasi *Pure Tank*

Definisi: Hero yang atribut utamanya 'Strength' DAN memiliki peran 'Durable'.

Aturan Prolog:

```
is_pure_tank(Hero) :-
    hero(Hero),
    is_strength_hero(Hero),          % Cek TBox
    is_durable(Hero).                % Cek TBox
```

Query:

```
| ?- is_pure_tank(X).

X = abyssal_underlord ? a

X = alchemist

X = axe

X = bristleback

X = centaur

X = chaos_knight

X = dawnbreaker

X = doom_bringer

X = dragon_knight

X = earth_spirit

X = elder_titan

X = huskar
```

Hasil ini sesuai dengan ekspektasi, di mana hero seperti Axe dan Centaur yang dikenal sebagai *tank* berhasil diklasifikasikan dengan benar.

Contoh 2: Klasifikasi *Glass Cannon*

Definisi: Hero yang perannya 'Nuker' ATAU 'Carry', tetapi BUKAN 'Durable'.

Aturan Prolog:

```
is_glass_cannon(Hero) :-
    hero(Hero),
    (is_nuker(Hero) ; is_carry(Hero)),
    \+ is_durable(Hero).
```

Query:

```
| ?- is_glass_cannon(X).

X = ancient_apparition ? a

X = antimage

X = antimage

X = arc_warden

X = arc_warden

X = bloodseeker

X = bloodseeker

X = bounty_hunter

X = broodmother

X = broodmother

X = clinkz

X = crystal_maiden

X = dark_willow

X = dazzle

X = death_prophet

X = death_prophet

X = disruptor
```

Aturan ini berhasil mengidentifikasi hero-hero yang memiliki kapabilitas serangan tinggi namun pertahanan rendah, seperti Bounty Hunter dan Clinkz.

Contoh 3: Klasifikasi *Magic Nuker*

Definisi: Hero dengan peran 'Nuker' yang memiliki setidaknya satu kemampuan aktif (bukan pasif) dengan tipe kerusakan 'Magical'.

Aturan Prolog:

```
is_magic_nuker(Hero) :-
    hero(Hero),
    is_nuker(Hero),                % Cek role 'Nuker' (dari TBox)
    has_ability(Hero, Ability),    % Temukan ability yang dia miliki (dari
    ABox)
    damage_type(Ability, magical), % Cek apakah damage-nya 'Magical' (dari
    ABox)
    ability_type(Ability, Tipe),   % Cek tipe ability-nya (dari ABox)
    Tipe \= passive.               % \= artinya 'tidak sama dengan'
```

Query:

```
| ?- is_magic_nuker(X).

X = abyssal_underlord ? a

X = abyssal_underlord

X = abyssal_underlord

X = abyssal_underlord

X = ancient_apparition

X = ancient_apparition

X = ancient_apparition

X = ancient_apparition

X = ancient_apparition

X = ancient_apparition

X = antimage

X = antimage

X = arc_warden

X = arc_warden

X = arc_warden
```

Query ini berhasil menemukan semua hero yang memenuhi kriteria *magic nuker*, seperti antimage dan abyssal underlord, dengan benar.

Hasil dari berbagai pengujian ini memvalidasi bahwa sistem berbasis pengetahuan kami berfungsi sesuai rancangan dan mampu melakukan penalaran klasifikasi secara akurat berdasarkan basis pengetahuan yang diekstrak dari ontologi.

4.2. Keterlibatan Logika Deskripsi

Meskipun kami tidak menulis aksioma DL secara manual, proses pembangunan ontologi kami secara inheren menggunakan konsep-konsep Logika Deskripsi:

- **Konsep (Concepts):** Hero, Ability, StrengthHero adalah contoh konsep.
- **Subsumsi (Subsumption):** StrengthHero $\sqsubseteq$ Hero (StrengthHero adalah subsums diari Hero).
- **Peran (Roles):** hasAbility, hasRole adalah contoh peran (properti objek).
- **Domain dan Range:** Kami mendefinisikan domain dan range untuk properti, contohnya domain(hasRole) = Hero dan range(hasRole) = Role.
- **Asersi (Assertions):** Setiap fakta dalam ABox, seperti hasRole(axe, initiator), adalah sebuah asersi yang menyatakan keanggotaan individu dalam suatu relasi.

Konversi ke Prolog pada dasarnya menerjemahkan asersi ini menjadi predikat logika orde pertama, yang memungkinkan kita untuk melakukan penalaran.

4.3. Keterbatasan

Ontologi:

- **Statis:** Ontologi kami hanya memodelkan data statis dan tidak dapat merepresentasikan keadaan permainan yang dinamis (misalnya, level Hero saat ini, item yang dibeli, atau efek *buff/debuff* yang sedang aktif).
- **Keterbatasan Relasi:** Relasi yang lebih kompleks seperti "kemampuan X meng-*counter* kemampuan Y" atau "item Z bersinergi baik dengan Hero A" tidak dimodelkan karena membutuhkan pengetahuan heuristik yang lebih mendalam.
- **Hierarki Datar di Prolog:** Skrip konversi aboxconvertprolog.py menyederhanakan hierarki kelas. Sebagai contoh, baik StrengthHero maupun AgilityHero hanya menjadi fakta hero(X). di Prolog. Informasi subkelas ini harus direkonstruksi melalui *query* yang melibatkan atribut utama.
- **Versi:** 7.39e, semua data pada ontologi dibuat berdasarkan dota 2 patch 7.39e.

Sistem Berbasis Pengetahuan:

- **Penalaran Sederhana:** KBS kami hanya mampu melakukan *query* dan pencocokan pola sederhana. Sistem ini tidak dapat memberikan rekomendasi strategis (misalnya, "pilih Hero apa untuk melawan tim lawan?") karena tidak memiliki aturan inferensi yang kompleks untuk itu.
- **Integrasi Manual:** Proses konversi dari OWL ke Prolog memerlukan skrip perantara. Seperti yang disebutkan dalam catatan kaki spesifikasi tugas, integrasi langsung antara RDF dan Prolog bisa jadi rumit, sehingga kami memilih pendekatan konversi ini sebagai solusi praktis.

5. Kesimpulan

Dalam proyek ini, kami telah berhasil merancang dan mengimplementasikan sebuah ontologi untuk domain Dota 2. Menggunakan data terstruktur dari file JSON, kami secara terprogram membangun ontologi OWL yang mendefinisikan kelas-kelas, properti, dan individu yang relevan. Selanjutnya, kami berhasil mengembangkan sebuah sistem berbasis pengetahuan dengan mengekstrak ABox dari ontologi dan menambahkan berbagai fakta yang didasarkan pada file ontology kemudian memasukkannya ke dalam Prolog.

Sistem Prolog yang dihasilkan mampu melakukan penalaran dan klasifikasi sederhana, seperti mencari Hero berdasarkan peran dan atribut, yang membuktikan bahwa representasi pengetahuan formal yang kami buat dapat dimanfaatkan secara praktis. Proyek ini memberikan kami pengalaman berharga dalam seluruh alur kerja sistem berbasis pengetahuan, mulai dari pemodelan domain, pembangunan ontologi, hingga implementasi sistem penalaran simbolik.

6. Lampiran

No.	Nama Mahasiswa	NIM	Deskripsi Kontribusi
1.	Sa’ad Abdul Hakim	13522092	ABox TBox KBS Rules Laporan
2.	Rayhan Fadhlan Azka	13522095	Scrape Data Pembuatan Ontologi TBox Laporan
3.	Rayendra Althaf Taraka Noor	13522107	Scrape TBox KBS Laporan